

UNIVERSIDAD DE CÁDIZ

FACULTAD DE CIENCIAS

GRADO EN MATEMÁTICAS

# MÉTODOS PARALELOS EN TIEMPO PARA ECUACIONES EN DERIVADAS PARCIALES

Noelia Ortega Román

Tutor: Dr. J. Rafael Rodríguez Galván





UNIVERSIDAD DE CÁDIZ

FACULTAD DE CIENCIAS

GRADO EN MATEMÁTICAS

# MÉTODOS PARALELOS EN TIEMPO PARA ECUACIONES EN DERIVADAS PARCIALES

Noelia Ortega Román

Tutor: Dr. J. Rafael Rodríguez Galván

Firma de la alumna



Noelia Ortega Román

Firma del tutor



J. Rafael Rodríguez Galván

Puerto Real, Cádiz, Julio de 2020





## **Abstract**

This essay is devoted to the study of time parallel numerical methods. Specifically, we focus on the Parareal algorithm, proposed for ordinary differential equations and which is here extended to partial differential equations.

After explaining some previous concepts (time numerical methods for differential equations) and studying the Finite Elements Method for partial differential equations, the shooting methods are introduced, as the key idea for the multiple shooting method. Finally, the Parareal algorithm is presented, and applied to solve the heat equation. To conclude this study, two epidemiological methods are introduced, the classic SIR model and an innovative scheme with diffusive terms, in order to be solved with Parareal so that the expounded theory can be verified.

*A mis padres, Pedro y Puri, y mis hermanos, José Antonio y Pedro.*

## **Resumen**

Este trabajo se enfoca al estudio de métodos numéricos paralelos en tiempo. En concreto, nos centraremos en el algoritmo Parareal, propuesto para ecuaciones diferenciales ordinarias y que extenderemos a ecuaciones en derivadas parciales.

Tras exponer los fundamentos previos necesarios (los métodos de resolución temporal para ecuaciones diferenciales) y estudiar el Método de los Elementos Finitos para la resolución espacial de ecuaciones en derivadas parciales, se introducen los métodos de disparo, como idea base del contenido posterior, los métodos de disparo múltiple. Finalmente, se presenta el algoritmo Parareal, y se aplica a la resolución de la ecuación del calor. Para concluir el estudio, se introducen dos modelos epidemiológicos, el modelo SIR clásico y un novedoso esquema con difusión, para ser resueltos con Parareal y comprobar la teoría expuesta.



## **Agradecimientos**

A mis profesores. A los que me han ido acompañando a lo largo de toda mi vida, aportando una parte esencial de lo que soy ahora. A aquellos que me han animado y exigido. Gracias por creer en mí, por formarme. En especial, gracias a todos los que hacen este grado posible, compartiendo su pasión por las matemáticas.

A mis compañeros y amigos. A los de siempre y a los de ahora. Gracias a los que me hicieron sentir cerca de ellos desde el primer día, estos cuatro años no habrían sido igual sin compartirlos con vosotros. Por el apoyo mutuo, la ayuda, las risas, las lágrimas, por vosotros.

A mi familia. Por ser pilar fundamental, no solo de este trabajo, sino de todo mi recorrido. Por ser las alas, el empujón y los brazos que me recogen. Por vuestra confianza puesta en mí desde que era una niña, animándome a llegar tan lejos como quisiera.

A mi profesor, tutor y amigo, Rafa. Por tu confianza en mí desde el primer momento. Por tu paciencia infinita, tus ánimos y tus ideas. Por nuestras risas, que aún pasando a ser virtuales, nunca desaparecieron. Por los cientos de nuevos proyectos que dibujamos en el aire en cada conversación. Gracias por hacer que este trabajo me ilusione, convirtiendo lo difícil en interesante, haciendo que siempre quiera un poco más.

Noelia Ortega

junio 2020



# Índice general

<b>1</b>	<b>Introducción</b>	<b>1</b>
<b>2</b>	<b>Motivación</b>	<b>5</b>
<b>3</b>	<b>Fundamentos teóricos</b>	<b>9</b>
3.1	Métodos numéricos . . . . .	9
3.2	Formulación variacional de EDP . . . . .	11
3.2.1	Espacios de Sobolev . . . . .	11
3.2.2	Formulación variacional . . . . .	15
3.3	El Método de los Elementos Finitos . . . . .	17
3.3.1	Convergencia . . . . .	19
3.4	El método de las líneas . . . . .	20
<b>4</b>	<b>Métodos de disparo</b>	<b>23</b>
4.1	Método de disparo . . . . .	23
4.2	Método de disparo múltiple en tiempo . . . . .	25
<b>5</b>	<b>Parareal</b>	<b>33</b>
5.1	Resultados teóricos . . . . .	35
5.2	Aproximación numérica de la ecuación del calor . . . . .	41
<b>6</b>	<b>Modelos epidemiológicos</b>	<b>47</b>
6.1	Modelos SIR . . . . .	47
6.2	Difusión . . . . .	53
<b>7</b>	<b>Conclusiones y objetivos futuros</b>	<b>61</b>

<b>A</b>	<b>Código Julia</b>	<b>65</b>
A.1	Ecuación del calor . . . . .	65
A.1.1	Solución exacta . . . . .	65
A.1.2	Solución numérica . . . . .	66
A.2	Modelo SIR . . . . .	70
A.3	Modelo SIR con difusión . . . . .	74
	<b>Bibliografía</b>	<b>81</b>



*This paper is dedicated to the proposition that, in order to take full advantage for real-time computations of highly parallel computers as can be expected to be available in the near future, much of numerical analysis will have to be recast in a more "parallel" form.*

J. Nievergelt

CAPÍTULO

# 1

## Introducción

La importancia de la modelización matemática en diversos campos, junto con los grandes avances tecnológicos de las últimas décadas, nos conducen a plantearnos de nuevo la idea de Nievergelt en 1964 [1]. Su propósito se basaba en la implementación de métodos de resolución paralelos que permitieran acelerar el cómputo de soluciones numéricas, un objetivo visionario, pues aún no existían ordenadores capaces de llevar a cabo dichos procesos. Ahora, varias décadas más tarde, nos proponemos hacer realidad esa idea, aplicándola a problemas actuales que requieren de una rápida resolución.

Un claro ejemplo en la necesidad de rapidez es el de los modelos epidemiológicos, cuya importancia ha destacado a lo largo de los últimos meses. La modelización de la evolución de la Covid-19 en la población puede venir dada por un modelo a gran escala que requiera de grandes tiempos de cómputo. Sin embargo, frente a una pandemia que afecta a gran parte del mundo, la actuación debe ser rápida y precisa. Por ello, centramos este trabajo en el estudio de técnicas de resolución paralelas en tiempo para ecuaciones en derivadas parciales (EDP), para finalmente aplicarlas a modelos de epidemias. Nuestro objetivo es comparar, en la medida de lo posible, estos métodos a los tradicionalmente secuenciales, con la intención de estudiar la mejora que presente uno frente a otro. Con un fin algo ambicioso, pretendemos conseguir una mejora lo mayor posible respecto a lo secuencial en problemas de resolución compleja.

## 1. INTRODUCCIÓN

---

Para aprovechar la capacidad de los nuevos potentes ordenadores, con diversos núcleos en los que realizar cálculos, históricamente se comenzó aplicando divisiones del problema en el dominio espacial, con técnicas de descomposición de dominio (consultar, por ejemplo, [2]). Sin embargo, cuando esta mejora es saturada, necesitamos nuevas formas de paralelización. Es aquí donde basamos nuestro trabajo, en una nueva técnica de resolución paralela, centrada no en el dominio espacial, si no en el temporal. De este modo, el cálculo de la solución se realizaría en diversos subproblemas, tantos como subintervalos temporales se consideren, los cuales serían resueltos de manera paralela. Dado que el paso del tiempo es secuencial, estos cálculos requieren de una corrección que mantenga la correlación entre un instante temporal y el siguiente. Así, estos procesos necesitan de una parte secuencial además de la paralela. Por ello, como mencionaba Nievergelt, este método exige mayor trabajo que un método secuencial clásico, pero, con la potencia computacional suficiente, obtendremos los resultados en un tiempo menor.

Comenzaremos con una breve motivación acerca del uso de la paralelización en nuestro trabajo, presentando un resultado clave, la ley de Amdahl. Posteriormente, introduciremos los conceptos necesarios para el estudio, comenzando por los métodos de aproximación en tiempo, Euler explícito e implícito, en los que se basará la aplicación de los algoritmos. En segundo lugar, es importante la introducción de la teoría de Análisis Funcional requerida para el Método de los Elementos Finitos, el cual será utilizado en la resolución espacial de los problemas propuestos. En esta sección serán presentados los espacios en los que se desarrolla la formulación variacional, así como los resultados principales de esta teoría y los subespacios de funciones polinómicas en los que tienen su base los elementos finitos. Será expuesto un teorema fundamental para la existencia y unicidad de solución para ciertos problemas, el teorema de Lax-Milgram. Finalmente, se estudiará la aplicación de los elementos finitos a problemas evolutivos.

Antes de la presentación del algoritmo paralelo en tiempo, es necesario introducir las ideas en las que se apoya. Comenzaremos, en el capítulo 4, con los métodos de disparo, una técnica basada en la resolución de un problema de contorno a partir de diversos problemas de valores iniciales. Cada uno de estos se resuelve partiendo de diferentes condiciones iniciales para la derivada, con el fin de encontrar aquella solución que satisfaga un valor

---

final concreto. A partir de esta idea, presentamos los métodos de disparo múltiple en tiempo. Estos métodos, aún secuenciales, tratan la resolución de problemas de valores iniciales mediante la división del dominio temporal en subintervalos, lo que nos va acercando ya al concepto en el que se fundamenta nuestro objetivo. Estas técnicas presentan buenas propiedades de convergencia, que serán heredadas en el capítulo siguiente.

En el capítulo 5 presentamos el algoritmo Parareal (desarrollado por J.-L. Lions, Y. Maday y G. Turinici en 2001 [3]), objeto principal de nuestro estudio. Comenzaremos explicando el procedimiento que lleva a cabo, para, posteriormente, exponer y demostrar tres resultados que tratan, respectivamente, su relación con lo anterior, convergencia y estimación de error. Tomaremos como ejemplo un problema cuya solución sea conocida, con el fin de poner a prueba el método. Para concluir este capítulo, compararemos los resultados con los de una resolución clásica secuencial.

Por último, en el capítulo 6, introduciremos los modelos epidemiológicos, a los que aplicaremos la teoría desarrollada previamente. Comenzaremos con un ejemplo sencillo, el modelo SIR, compuesto por un sistema de ecuaciones diferenciales. La relativa simplicidad de este sistema frente a las ecuaciones en derivadas parciales, nos permitirá hacer numerosas comprobaciones, dada la rapidez con la que se obtiene solución al no necesitar adaptaciones para conjugar el funcionamiento de los algoritmos paralelos en tiempo junto con las aproximaciones espaciales dadas por el Método de los Elementos Finitos. Finalmente, nos centramos en un modelo epidemiológico con difusión que hemos desarrollado expresamente para esta memoria, con el fin de comparar paralelización y técnicas clásicas secuenciales. Hasta donde conocemos, un modelo con estas características no ha sido publicado con anterioridad, adentrándonos, en este sentido, en la investigación.

Tan importante como el diseño de los algoritmos en este trabajo, es la elección del lenguaje de programación y la arquitectura paralela utilizada. Dado que se trata de un tema actualmente en estudio, es difícil encontrar un lenguaje que soporte los diferentes requisitos del programa para la resolución. Los algoritmos paralelos que hemos estudiado están orientados a ecuaciones diferenciales ordinarias o, a lo sumo, a ecuaciones en derivadas parciales mediante el método de las diferencias finitas. En este trabajo, se va más allá,

## 1. INTRODUCCIÓN

---

aplicando en la resolución espacial el Método de los Elementos Finitos, cuyas bibliotecas no suelen ser soportadas en lenguajes previamente conocidos, como `Python`, por las necesarias para la paralelización. Por ello, se opta por `Julia`, un lenguaje dirigido a la computación científica y numérica, compilado en tiempo de ejecución. El hecho de que sea un lenguaje diseñado para el paralelismo, hace posible la implementación de programas de resolución que hagan uso del Método de los Elementos Finitos, gracias a bibliotecas como `Gridap` [4]. Por último, cabe nombrar el visualizador 3D utilizado para la implementación de las gráficas, `Paraview`, que nos hace posible la representación de modelos con múltiples incógnitas. El código empleado en este estudio se encuentra a disposición de quien lo desee, además de en el apéndice A, en el repositorio de `GitHub`:

`https://github.com/NoeliaOrtega/Julia.git`

# Motivación

En las últimas décadas, el gran avance computacional ha llevado a la búsqueda de nuevas técnicas de resolución que permitan obtener soluciones a grandes problemas en cortos periodos de tiempo.

Siguiendo con la idea de las ecuaciones en derivadas parciales, planteémonos la cuestión de resolver un sistema donde la partición del dominio presenta  $N$  partes. En la resolución clásica, cada una de estas parte se resolvería de manera secuencial, pues es cierto que se requiere de un proceso iterativo al estar cada subdominio influenciado por los restantes. Pero, ¿y si pudiéramos resolver cada uno de estos subdominios de manera paralela? Podríamos aprovechar los núcleos de los nuevos potentes ordenadores para disminuir el tiempo de computo de manera muy significativa.

Esta se ha convertido en una idea fundamental en los últimos años, donde se ha podido comprobar que la complejidad de los macroprocesadores ha alcanzado su límite y la ley de Moore (ley informática que pronosticaba que cada dos años se duplicaría el número de transistores de un macroprocesador) ha dejado de ser válida. La única solución para aumentar significativamente la potencia de los ordenadores es aumentar el número de procesadores paralelos.

Aunque la idea parece buena, se observó rápidamente que existe un límite para el paralelismo disponible en un ordenador. Este límite viene dado por el porcentaje de parte secuencial necesaria a la hora de llevar a cabo un proceso. Esto se conoce como la ley de

## 2. MOTIVACIÓN

---

Amdahl. Para más información acerca de la computación paralela, se aconseja consultar, por ejemplo, [5].

**Proposición 2.1** (*Ley de Amdahl*). Sea  $O_p$  el número de operaciones que lleva a cabo un programa paralelo en  $p$  procesadores. Si la porción  $f_p$ , con  $0 < f_p < 1$ , de las  $O_p$  operaciones es necesariamente secuencial, entonces el incremento en la velocidad está acotado por

$$S_p < \frac{1}{f_p}.$$

*Demostración.* En primer lugar, siendo  $p$  el número de procesadores de un ordenador, consideremos:

- $T_p$ : número de pasos del algoritmo paralelo en  $p$  procesadores, donde cada paso consiste en:
  - La comunicación entre procesadores y las referencias a la memoria.
  - Las operaciones aritméticas llevadas a cabo.
- $T_1$ : número de operaciones aritméticas necesarias en el algoritmo secuencial.

De estas definiciones, es obvio que

$$T_p \geq \left( f_p + \frac{1 - f_p}{p} \right) T_1.$$

Por otra parte, dado que  $S_p$  representa el incremento en la velocidad del proceso paralelo frente al secuencial, vendrá dado por:

$$S_p = \frac{T_1}{T_p}$$

Haciendo uso de ambos resultados, tenemos que

$$S_p = \frac{T_1}{T_p} \leq \frac{1}{\left( f_p + \frac{1 - f_p}{p} \right)} < \frac{1}{f_p}, \quad \text{ya que} \quad \frac{1 - f_p}{p} > 0.$$

□

Es decir, la mejora que obtenemos al paralelar no es infinita, de modo que si, partiendo de los  $N$  subdominios, tenemos  $N$  núcleos, hemos saturado la mejora. Cada núcleo resolverá un subproblema y seguiremos teniendo un gran desgaste temporal en la conexión entre las distintas soluciones. Es por esto que nos planteamos la cuestión de cómo seguir explotando la capacidad de la que se dispone computacionalmente. Habitualmente, las técnicas

---

de paralelización para la resolución numérica de EDP se basan en dividir el dominio espacial en regiones sobre las que se plantean los subproblemas. Introducimos aquí el objetivo de paralelar no solo el espacio, si no también el tiempo. Sin embargo, esta idea ya surgió hace casi 50 años.

En 1964, J. Nievergelt [1] fue el primero en considerar una descomposición en tiempo pura del dominio. Ya entonces tuvo la visionaria idea de que, en un futuro, la tecnología alcanzaría mejoras que nos permitirían reducir significativamente los tiempos de computo. Su objetivo era paralelar la integración numérica de una EDO, un proceso secuencial. El método consistía en dividir el intervalo temporal en subintervalos y aplicar un método paralelo en tiempo de resolución.

Décadas después nos planteamos llevar a cabo dicha idea, pues, con los avances tecnológicos actuales, este hecho proporcionaría grandes ahorros de tiempo computacional que serían de gran interés e importancia en áreas que requieren de las matemáticas. Un claro ejemplo son los modelos de epidemias para los que, aún tratándose de forma tradicional con sistemas de ecuaciones diferenciales, se ha empezado a investigar la introducción de términos de difusión y en general, de ecuaciones en derivadas parciales. Encontramos así la motivación de dar respuesta a la evolución de una población afectada por una enfermedad, no solo indicando el número de individuos contagiados, sino también, dónde se producirán dichos contagios.





# Fundamentos teóricos

En este apartado se expondrá de manera resumida la teoría necesaria para llevar a cabo el estudio de los métodos paralelos. Para ello, se planteará como fundamento previo el método de Euler, seguido por una introducción al Método de los Elementos Finitos y los conceptos en los que se basa. Por último, se estudiará cómo compaginar los elementos finitos con ecuaciones evolutivas. Para ello, se presentará el método de las líneas, como una forma de transformar el problema discreto en un sistema de ecuaciones diferenciales ordinarias.

## 3.1 Métodos numéricos

Cuando se tratan ecuaciones que modelan la realidad, los resultados suelen no ser tan sencillos como en los ejemplos teóricos. Es usual encontrar ecuaciones cuya solución sea muy complicada o imposible de expresar en término de funciones elementales. Por ello, se utilizan los métodos numéricos, que permiten transformar el proceso de resolución en operaciones aritméticas. Estas proporcionan soluciones aproximadas para las que, utilizando los resultados teóricos del análisis numérico, se puede garantizar la convergencia hacia la solución exacta. En este apartado se hablará de aquellos utilizados en el estudio posterior.

Como métodos básicos para la aproximación de soluciones a las ecuaciones en derivadas parciales en el dominio temporal, nos centraremos en los métodos de *Euler explícito* e

### 3. FUNDAMENTOS TEÓRICOS

---

*implícito.*

En general, el método de Euler consiste en definir una sucesión de soluciones,  $\{y_n\}_n$ , en una partición del dominio en el que se encuentre la ecuación. Estas soluciones se calculan de manera iterativa partiendo de un dato inicial conocido  $y_a$ .

Considérese una función  $f$  continua en  $\Omega \times [a, b]$  y una partición del intervalo  $[a, b]$  formada por  $N + 1$  puntos:

$$a = t_0 < t_1 < \dots < t_N = b.$$

Por simplicidad, supondremos que la partición es uniforme, siendo el tamaño de cada subintervalo  $k$ , con  $k = \frac{b-a}{N}$ . El método de Euler explícito sigue el esquema:

$$\begin{cases} y_0 = y_a, \\ y_{n+1} = y_n + kf(y_n, t_n), \quad n = 0, \dots, N-1 \end{cases}$$

En el caso de Euler implícito:

$$\begin{cases} y_0 = y_a, \\ y_{n+1} = y_n + kf(y_{n+1}, t_{n+1}), \quad n = 0, \dots, N-1 \end{cases}$$

A continuación se muestran algunos resultados teóricos que garantizan la convergencia del método de Euler. Las demostraciones pueden encontrarse, por ejemplo, en [6, 7].

**Notación 3.1.** Para cualquier sucesión  $\{y_n\}_n$ , se utilizará la siguiente notación para el cociente incremental:

$$\delta_t y_{n+1} = \frac{y_{n+1} - y_n}{k}.$$

Supondremos que la solución exacta viene dada por  $g(t)$ .

**Definición 3.1.** Para cada  $n = 0, 1, \dots, N-1$ , se define el error de consistencia del método de Euler en el instante  $t_n$  como

$$\mathcal{E}_n = \delta_t g(t_{n+1}) - f(t_n, g(t_n)).$$

**Teorema 3.1. (Consistencia)** Si la solución exacta del problema es  $g \in \mathcal{C}^2([a, b])$ , entonces el método de Euler es consistente. Más aún, se verifica

$$\|\varepsilon(k)\|_\infty \leq \frac{k}{2} \|g''\|_\infty,$$

por tanto, el orden de consistencia del método de Euler es igual a uno.

**Teorema 3.2.** (*Estabilidad*) Si  $f$  es globalmente  $y$ -Lipschitz (con constante  $L$ ), entonces el método de Euler es estable. Más aún, se verifica la siguiente estimación de estabilidad:

$$\|z_n - y_n\|_\infty \leq C_1|\mu_0| + C_2\|\delta_n\|_\infty,$$

donde  $\delta_n$  es la perturbación producida en la iteración  $n$ ,  $C_1 = e^{(b-a)L}$ ,  $C_2 = \frac{e^{(b-a)L}-1}{L}$  y  $\mu_0$  denota a un posible error en el dato inicial.

**Teorema 3.3.** Supongamos que  $f$  es  $y$ -Lipschitz. Entonces, se verifica la siguiente estimación del error de discretización:

$$\|E\|_\infty \leq C_1|e_0| + C_2\|\varepsilon\|_\infty,$$

donde  $C_1$  y  $C_2$  viene dados en el teorema anterior, y  $E = E(h) = (e_0, e_1, \dots, e_n)$  es el error de discretización.

**Corolario 3.1.** (*Convergencia y orden del método de Euler*) Si  $f$  es  $y$ -Lipschitz y además  $g \in \mathcal{C}^2([a, b])$ , entonces el método de Euler es convergente y tiene orden 1. En concreto, si  $|e_0| \leq C_0 h$  para alguna constante  $C_0 \geq 0$ , entonces:

$$\|E\|_\infty \leq C_0 C_1 h + C_2 \|g''\|_\infty \frac{h}{2}.$$

## 3.2 Formulación variacional de EDP

En esta sección expondremos brevemente los conceptos matemáticos en los que tiene su base el Método de los Elementos Finitos, el cual será utilizado en este estudio para la resolución espacial de los problemas. Información más detallada puede encontrarse, por ejemplo, en [8, 9, 10, 11].

### 3.2.1 Espacios de Sobolev

Comenzamos presentando los espacios de Sobolev. Dichos espacios de funciones serán la base para la resolución numérica de EDP mediante el Método de los Elementos Finitos, pues en ellos se definirá la formulación variacional, presentada en el posterior apartado.

**Definición 3.2.** Sea  $\Omega \subseteq \mathbb{R}^N$  abierto. Para cada  $p \geq 1$  se define el siguiente espacio:

$$L^p(\Omega) = \{f : \Omega \rightarrow \mathbb{R} : f \text{ es medible Lebesgue y } \int_\Omega |f|^p dx < \infty\}.$$

Para  $p = +\infty$ , definimos

$$L^\infty = \{f : \Omega \rightarrow \mathbb{R} : f \text{ medible Lebesgue y } \exists c > 0, |f(x)| \leq c \text{ casi por doquier, } x \in \Omega\}.$$

### 3. FUNDAMENTOS TEÓRICOS

---

Es especialmente interesante el caso concreto de  $p = 2$ .

**Observación 3.1.** En el espacio  $L^2(\Omega)$  se considera el producto escalar

$$(u, v)_{L^2(\Omega)} = \int_{\Omega} u(x)v(x)dx, \quad u, v \in L^2(\Omega).$$

Y la norma asociada a este producto

$$\|u\|_{L^2(\Omega)} = \sqrt{(u, u)} = \left( \int_{\Omega} u^2(x)dx \right)^{\frac{1}{2}}, \quad u \in L^2(\Omega).$$

**Lema 3.1.** Sean  $u, v \in L^2(\Omega)$ . Entonces  $uv \in L^1(\Omega)$  y, de hecho, se verifica la desigualdad de Cauchy-Schwarz:

$$\int_{\Omega} |u(x)v(x)|dx \leq \left( \int_{\Omega} u^2(x)dx \right)^{1/2} \left( \int_{\Omega} v^2(x)dx \right)^{1/2}$$

*Demostración.* La desigualdad de Cauchy-Schwarz es un resultado clásico cuya demostración puede encontrarse, por ejemplo, en [10] (capítulo 2). El resto del lema es consecuencia directa. Así, consideremos  $u, v \in L^2(\Omega)$  y veamos que el producto de estas funciones pertenece al espacio  $L^1(\Omega) = \{f : \Omega \rightarrow \mathbb{R}, \text{medibles Lebesgue}, \int_{\Omega} |f|dx < \infty\}$ . Aplicando la desigualdad de Cauchy-Schwarz:

$$\int_{\Omega} |u(x)v(x)|dx \leq \left( \int_{\Omega} u^2(x)dx \right)^{1/2} \left( \int_{\Omega} v^2(x)dx \right)^{1/2}$$

Por definición de  $L^2(\Omega)$ , tenemos que cada una de estas integrales son finitas. Luego, el producto de ambas es finito y  $uv \in L^1(\Omega)$ .  $\square$

**Definición 3.3.** Sea  $v$  una función de  $L^p(\Omega)$  y sea  $\mathcal{C}_c^\infty$  el espacio de las funciones de clase  $\mathcal{C}^\infty$  con soporte compacto en  $\Omega$ . Decimos que  $v$  es diferenciable en sentido débil en  $L^p(\Omega)$  si existen funciones  $w_i \in L^p(\Omega)$ , para  $i \in \{1, \dots, N\}$ , tales que, para cada función  $\phi \in \mathcal{C}_c^\infty(\Omega)$ , se tiene

$$\int_{\Omega} v(x) \frac{\partial \phi}{\partial x_i}(x)dx = - \int_{\Omega} w_i(x) \phi(x)dx.$$

Cada  $w_i$  se denomina  $i$ -ésima derivada parcial débil de  $v$  y se denota por  $\frac{\partial v}{\partial x_i}$ .

**Definición 3.4.** Sea  $\Omega \subset \mathbb{R}^N$  abierto. Un espacio de Sobolev  $W^{m,p}(\Omega)$  ( $m \in \mathbb{N} \cup \{0\}$ ,  $p \geq 1$ ) es el espacio lineal de funciones dentro de  $L^p(\Omega)$  cuyas derivadas (en sentido débil) de orden menor o igual a  $m$  están en  $L^p$ . Entonces,

$$W^{m,p}(\Omega) = \{v \in L^p(\Omega) \mid \int_{\Omega} |D^{\alpha}v|^p dx < \infty \text{ para } 0 \leq |\alpha| \leq m\},$$

donde  $D^{\alpha}v = \frac{\partial^{|\alpha|}v}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}}$  y  $|\alpha| = \alpha_1 + \alpha_2 + \dots + \alpha_n$ .  
Para  $m = 0$ ,  $W^{0,p}(\Omega) = L^p(\Omega)$ .

Los espacio de Sobolev, generalmente, son dotados se las siguientes normas:

$$\|v\|_{m,p,\Omega} = \left[ \int_{\Omega} \sum_{0 \leq |\alpha| \leq m} |D^{\alpha}v|^p dx \right]^{\frac{1}{p}} = \left[ \sum_{0 \leq |\alpha| \leq m} \|D^{\alpha}v\|_{L^p(\Omega)}^p \right]^{\frac{1}{p}}, \quad \text{para } 0 \leq p < m, \quad (3.1)$$

$$\|v\|_{m,\infty,\Omega} = \max_{0 \leq |\alpha| \leq m} \|D^{\alpha}v\|_{L^{\infty}(\Omega)}. \quad (3.2)$$

**Observación 3.2.** Para  $p = 2$  denotamos  $H^m(\Omega) = W^{m,2}(\Omega)$ . Se puede demostrar (ver, por ejemplo, [12]) que para todo  $m \geq 0$  los espacios  $H^m(\Omega)$  son espacios de Hilbert. Estos espacios están dotados con los productos escalares dados por

$$(u, v)_{m,\Omega} = \int_{\Omega} \sum_{0 \leq |\alpha| \leq m} D^{\alpha}u D^{\alpha}v dx.$$

Sus normas asociadas está dadas por

$$\|v\|_{m,\Omega} = \|v\|_{m,2,\Omega} = [(v, v)_{m,\Omega}]^{\frac{1}{2}}.$$

**Definición 3.5.** El espacio de Sobolev  $H_0^1(\Omega)$  se define como la clausura de  $\mathcal{C}_c^{\infty}(\Omega)$  en  $H^1(\Omega)$ .

Para poder enunciar un resultado que nos facilitará la compresión del espacio  $H_0^1(\Omega)$ , necesitamos la siguiente definición:

**Definición 3.6.** Sea  $\Omega \subset \mathbb{R}^N$  abierto, regular y acotado, de clase  $\mathcal{C}^1$ . Se define la aplicación traza como

$$\begin{aligned} \gamma : H^1(\Omega) \cap \mathcal{C}(\overline{\Omega}) &\rightarrow L^2(\partial\Omega) \cap \mathcal{C}(\partial\overline{\Omega}) \\ u &\mapsto \gamma(u) = u|_{\partial\Omega}. \end{aligned}$$

### 3. FUNDAMENTOS TEÓRICOS

---

**Observación 3.3.** Se dice que un abierto es de clase  $\mathcal{C}^k$  ( $k \geq 1$ ) si su frontera se puede parametrizar por funciones que son localmente de clase  $\mathcal{C}^k$ . Ver, por ejemplo, [8] definición 3.2.5.

Se puede demostrar ([12]) que esta aplicación  $\gamma$  se extiende por continuidad a una aplicación lineal y continua de  $H^1(\Omega)$  en  $L^2(\partial\Omega)$  también llamada  $\gamma$ . En particular, existe una constante  $C > 0$  tal que para cada función  $u \in H^1(\Omega)$  tenemos que

$$\|\gamma(u)\|_{L^2(\partial\Omega)} \leq C\|u\|_{H^1(\Omega)}.$$

Así, considerando que  $u|_{\partial\Omega}$  hace referencia a la aplicación traza, tenemos que:

**Proposición 3.1.** Sea  $\Omega$  un conjunto abierto, regular y acotado de clase  $\mathcal{C}^1$ . El espacio  $H_0^1(\Omega)$  coincide con el subespacio  $H^1(\Omega)$  formado por las funciones que son cero en la frontera  $\partial\Omega$  (es decir, que tienen traza cero en  $\partial\Omega$ ).

Por último, vamos a presentar dos resultados que nos serán de utilidad en las siguientes secciones.

**Teorema 3.4 (Fórmula de Green).** Sea  $\Omega \in \mathbb{R}^N$  un conjunto abierto, regular y acotado de clase  $\mathcal{C}^1$ . Sea  $w \in \mathcal{C}^1(\overline{\Omega})$ , con soporte contenido en la clausura  $\overline{\Omega}$ . Entonces,  $w$  satisface la fórmula de Green

$$\int_{\Omega} \frac{\partial w}{\partial x_i}(x) dx = \int_{\partial\Omega} w(x) n_i(x) ds,$$

donde  $n_i, i \in \{1, \dots, N\}$  es la  $i$ -ésima componente del vector normal unitario apuntando hacia fuera de  $\partial\Omega$ .

Como consecuencia a este teorema, tenemos el siguiente corolario que nos será de gran utilidad:

**Corolario 3.2 (Fórmula de integración por partes).** Sea  $\Omega \in \mathbb{R}^N$  un conjunto abierto y regular, de clase  $\mathcal{C}^1$ . Sean  $u, v \in \mathcal{C}^1(\overline{\Omega})$  dos funciones con soporte contenido en el conjunto cerrado  $\overline{\Omega}$ . Entonces, estas funciones satisfacen la fórmula de integración por partes

$$\int_{\Omega} u(x) \frac{\partial v}{\partial x_i}(x) dx = - \int_{\Omega} v(x) \frac{\partial u}{\partial x_i}(x) dx + \int_{\partial\Omega} u(x) v(x) n_i(x) ds,$$

donde  $n_i, i \in \{1, \dots, N\}$  es la  $i$ -ésima componente del vector normal unitario apuntando hacia fuera de  $\partial\Omega$ .

### 3.2.2 Formulación variacional

El objetivo de este apartado es mostrar la relación entre la formulación clásica de un problema y la formulación variacional. Esta última es la que se utiliza a la hora de aplicar el Método de los Elementos Finitos.

Consideremos el problema de contorno:

$$\begin{cases} -\Delta u = f & \text{en } \Omega \\ u = 0 & \text{en } \partial\Omega \end{cases} \quad (3.3)$$

La **formulación clásica** se basa en asumir que la solución al problema es suficientemente regular; se supone que la solución tiene sentido en cada punto del espacio. De este modo, se llama **solución clásica** a toda función  $u \in \mathcal{C}^2(\Omega) \cap \mathcal{C}(\partial\Omega)$  que verifique (3.3) en cada punto de  $\Omega$  y  $\partial\Omega$ .

El problema de esta formulación es su fuerte hipótesis sobre regularidad de la solución. En general, no siempre podremos asegurar que existan soluciones de las EDP consideradas tales que dicha función sea de clase  $\mathcal{C}^2$ . Por este motivo introducimos la formulación variacional o formulación débil, cuyo objetivo es convertir nuestro problema en otro tal que la solución no necesite condiciones tan restrictivas.

Partiendo del problema (3.3), la idea que se va a plantear es encontrar  $u \in V = H_0^1(\Omega)$  tal que

$$-\int_{\Omega} \Delta u v = \int_{\Omega} f v.$$

Con este fin, presentamos el siguiente resultado.

**Proposición 3.2.** Sea  $\Omega$  un conjunto abierto, acotado y regular. Supongamos  $f$  en (3.3) continua. Sea  $u$  una función de clase  $\mathcal{C}^2(\overline{\Omega})$ . Sea  $X$  el espacio definido por

$$X = \{\phi \in \mathcal{C}^1(\overline{\Omega}) \text{ tal que } \phi = 0 \text{ en } \partial\Omega\}.$$

Entonces  $u$  es solución del problema de contorno (3.3) si y solo si  $u$  pertenece a  $X$  y satisface la ecuación

$$\int_{\Omega} \nabla u(x) \nabla v(x) dx = \int_{\Omega} f(x) v(x) dx, \quad \forall v \in X. \quad (3.4)$$

La ecuación (3.4) se denomina **formulación variacional** del problema (3.3).

### 3. FUNDAMENTOS TEÓRICOS

---

*Demostración.* Si  $u$  es solución del problema de contorno (3.3), multiplicamos la ecuación  $-\Delta u = f$  por  $v \in X$  y como consecuencia del corolario 3.2:

$$\int_{\Omega} \Delta u(x) v(x) dx = - \int_{\Omega} \nabla u(x) \cdot \nabla v(x) dx + \int_{\partial\Omega} (\nabla u(x) \cdot \vec{n}) v(x) ds$$

Como  $v = 0$  en  $\partial\Omega$ , tenemos que el segundo término de la derecha es nulo, luego

$$\int_{\Omega} f(x) v(x) dx = - \int_{\Omega} \nabla u(x) \cdot \nabla v(x) dx$$

Por otro lado, si  $u \in X$  satisface (3.4), aplicando ahora la fórmula inversa de integración por partes, tenemos que

$$\int_{\Omega} (\Delta u(x) + f(x)) v(x) dx = 0 \quad \forall v \in X$$

Como  $(\Delta u + f)$  es una función continua, tenemos que  $-\Delta u(x) = f(x)$  para todo  $x \in \Omega$ . Además, como  $u \in X$ , tenemos que se verifica la condición de frontera, es decir,  $u = 0$  en  $\partial\Omega$ . Luego,  $u$  es solución del problema de contorno (3.3).  $\square$

De esta forma hemos conseguido transformar la formulación clásica (3.3) en una formulación más débil (3.4), para la que podríamos esperar que fuera más fácil demostrar la existencia de solución. Por ejemplo, para que (3.4) tenga sentido, bastaría que  $u \in H^1(\Omega)$  y  $f \in L^2(\Omega)$ . Y, de hecho, se tiene el siguiente resultado:

**Teorema 3.5.** Sea  $\Omega$  un conjunto abierto y acotado de  $\mathbb{R}^N$ . Consideremos  $f \in L^2(\Omega)$ . Entonces, existe una única solución  $u \in H_0^1(\Omega)$  solución de la formulación variacional (3.4). Además,  $u$  satisface

$$-\Delta u = f \text{ en casi todo } \Omega.$$

Si suponemos que  $\Omega$  es regular de clase  $\mathcal{C}^1$ , entonces  $u$  es solución del problema de contorno (3.3) en el sentido de

$$-\Delta u = f \text{ en casi todo } \Omega, \quad u = 0 \text{ en casi todo } \partial\Omega.$$

Entonces, nuestra pregunta ahora es, si  $u \in H^1(\Omega)$  y  $u$  verifica (3.4), ¿implica esto que  $u$  sea también solución de (3.3)? Dicho de otra forma, ¿la formulación variacional implica la formulación clásica? La respuesta es sí, bajo ciertas hipótesis. Por ejemplo, bastaría con que se verificara lo anterior y:

- $f \in \mathcal{C}^0(\Omega)$



- $\partial\Omega$  es regular.

Para más información acerca de estas afirmaciones, se recomienda consultar el apartado IX.6 de [12].

### 3.3 El Método de los Elementos Finitos

Tras haber introducido la formulación variacional, nos centramos ahora en el Método de los Elementos Finitos para aproximar la solución de la EDP (3.3). La idea de este método consiste en reemplazar el espacio de Hilbert,  $V$ , antes considerado, por uno de dimensión finita,  $V_h \subset V$ , cuyas funciones son polinómicas a trozos.

Sea  $V_h \subset V$  finito dimensional y con base  $\mathcal{B} = \{\varphi_1, \dots, \varphi_n\}$ , nuestro nuevo problema sería: encontrar  $u_h \in V_h$  tal que para todo  $v_h \in V_h$  se verifique

$$\int_{\Omega} \nabla u_h \nabla v_h = \int_{\Omega} f v_h,$$

donde  $u_h = \sum_{j=1}^n u_j \varphi_j$ , con  $u_j \in \mathbb{R}$ , y  $v_h = \sum_{i=1}^n v_i \varphi_i$ , con  $v_i \in \mathbb{R}$ .

Así, como debe cumplirse para cada  $v_h \in V_h$ , en concreto se cumple para los elementos de la base (aquellos  $v_h$  tales que  $v_k = 0$ , para todo  $k \neq i$  y  $v_i = 1$ ).

$$\sum_{j=1}^n u_j \int_{\Omega} \nabla \varphi_j \nabla \varphi_i = \int_{\Omega} f \varphi_i.$$

De este modo, resolver el problema se reduce a un sistema lineal, cuya matriz se denomina matriz de rigidez. Así, nuestro objetivo será construir  $V_h$  de manera que sea una buena aproximación de  $V$  y tal que la solución  $u_h$  sea también una buena aproximación de la solución  $u$  en  $V$ .

Denotando por  $a(u, v)$  a la forma bilineal del primer miembro de la desigualdad anterior, y por  $L(v)$  al segundo miembro, el problema se transforma en:

$$\text{encontrar } u_h \in V_h \text{ tal que } a(u_h, v_h) = L(v_h) \quad \text{para todo } v_h \in V_h \quad (3.5)$$

**Definición 3.7.** Sea  $X$  un espacio normado y  $B(\cdot, \cdot) : X \times X \rightarrow \mathbb{R}$  una forma bilineal. Se dice que  $B$  es:

### 3. FUNDAMENTOS TEÓRICOS

---

1. continua, si existe una constante  $\delta$  tal que

$$\|B(x_1, x_2)\| \leq \delta \|x_1\| \cdot \|x_2\|, \text{ para cada } (x_1, x_2) \in X_1 \times X_2.$$

2. coercitiva si existe una constante positiva  $\alpha$  tal que

$$B(x, x) \geq \alpha \|x\|^2, \quad \text{para todo } x \in X.$$

**Teorema 3.6** (Lax-Milgram). Sea  $V$  un espacio de Hilbert, y  $V_h$  un subespacio de dimensión finita. Sea  $a(u, v)$  una forma bilineal continua y coercitiva sobre  $V$ , y  $L(v)$  una forma lineal continua sobre  $V$ . Entonces, el problema (3.5) tiene una única solución. Además, esta solución puede ser obtenida mediante la resolución de un sistema lineal con una matriz definida positiva (y simétrica si  $a(u, v)$  es simétrico).

La demostración de este resultado puede hallarse, por ejemplo, en [8, 12].

En el proceso descrito hasta ahora, que es conocido como **Método de Galerkin**, no hemos especificado la naturaleza de los espacios finito-dimensionales  $V_h$ . En el Método de los Elementos Finitos, para construir los espacios de aproximación interna  $V_h$  de los espacios de funciones  $H^1(\Omega)$ ,  $H_0^1(\Omega)$ , ..., nos basamos en el concepto geométrico de malla del dominio  $\Omega$ .

**Definición 3.8.** Sea  $\Omega$  un poliedro abierto y conexo de  $\mathbb{R}^N$ . Una malla triangular es un conjunto  $\mathcal{T}_h$  de N-símplices  $(K_i)_{1 \leq i \leq n}$  que satisface

1.  $K_i \in \overline{\Omega}$  y  $\overline{\Omega} = \cup_{i=1}^n K_i$ .
2. La intersección de dos N-símplices distintos,  $K_i \cap K_j$ , es un m-símplice, con  $0 \leq m \leq N - 1$ , cuyos vértices son también vértices de  $K_i$  y  $K_j$ .

Los vértices o nodos de la malla  $\mathcal{T}_h$  son los vértices de los N-símplices que la componen. Por convenio, el parámetro  $h$  denota el máximo diámetro de los N-símplices  $K_i$ .

**Observación 3.4.** En el límite  $h \rightarrow 0$ ,  $V_h$  será cada vez más y más grande, aproximándose así al espacio entero  $V$ . Además, la matriz de rigidez  $\mathcal{K}_h$  del sistema lineal será dispersa; es decir, la mayoría de sus coeficientes serán nulos.

**Definición 3.9.** Sea  $K$  un N-símplice con vértices  $(a_j)_{1 \leq j \leq N+1}$ , las coordenadas baricéntricas  $(\lambda_j)_{1 \leq j \leq N+1}$  de  $x \in \mathbb{R}^N$  se definen por

$$\sum_{j=1}^{N+1} \lambda_j = 1, \quad \sum_{j=1}^{N+1} a_{i,j} \lambda_j = x_i, \quad \text{para } 1 \leq i \leq N.$$

**Definición 3.10.** Sea  $K$  un  $N$ -símplice, llamamos retículo de orden  $k$  al conjunto

$$\sum_k = \{x \in \mathbb{R}^N \text{ tales que } \lambda_j(x) \in \left\{0, \frac{1}{k}, \dots, \frac{k-1}{k}, 1\right\} \text{ para } 1 \leq j \leq N\}$$

donde  $\lambda_j, j = 1, \dots, N+1$ , son las coordenadas baricéntricas de  $x$ .

A continuación, van a presentarse dos resultados cuyas demostraciones puede consultarse, por ejemplo, en [8].

**Lema 3.2.** Sea  $K$  un  $N$ -símplice. Para un entero  $k \geq 0$ , sea  $\sum_k$  el retículo de orden  $k$ , cuyos puntos se denotan por  $(\sigma_j)_{1 \leq j \leq n_k}$ . Entonces, cada polinomio de  $\mathbb{P}_k$  esta determinado unívocamente por sus valores en los puntos  $(\sigma_j)_{1 \leq j \leq n_k}$ . Es decir, existe una base  $(\psi)_{1 \leq j \leq n_k}$  de  $\mathbb{P}_k$  tal que

$$\psi_j(\sigma_i) = \delta_{ij} \quad 1 \leq i, j \leq n_k.$$

**Definición 3.11.** Dada una malla  $\mathcal{T}_h$  de un poliedro abierto y conexo  $\Omega$ , se define el Método de los Elementos Finitos  $\mathbb{P}_k$  asociado a la malla  $\mathcal{T}_h$  como el Método de Galerkin al espacio discreto

$$V_h = \{v \in \mathcal{C}(\bar{\Omega}) \text{ tales que } v|_{K_i} \in \mathbb{P}_k \text{ para todo } K_i \in \mathcal{T}_h\}.$$

Llamamos grados de libertad de una función  $v \in V_h$  al conjunto de valores  $\{v(\bar{a}_i)\}_{1 \leq i \leq n_{dl}}$ , donde  $\{\bar{a}_i\}_{1 \leq i \leq n_{dl}}$  es el conjunto de puntos del retículo de orden  $k$  de cada  $N$ -símplice  $K_i \in \mathcal{T}_h$ , y donde  $n_{dl}$  es el número de grados de libertad.

También definimos el subespacio  $V_{0h}$  como

$$V_{0h} = \{v \in V_h \text{ tales que } v = 0 \text{ en } \partial\Omega\}.$$

**Proposición 3.3.** El espacio  $V_h$  definido anteriormente, es un subespacio de  $H^1(\Omega)$  cuya dimensión es finita e igual al número de grados de libertad. Además, existe una base de  $V_h$ ,  $(\phi_i)_{1 \leq i \leq n_{dl}}$  definida como

$$\phi_i(\bar{a}_j) = \delta_{ij} \quad 1 \leq i, j \leq n_{dl}, \text{ tal que } v(x) = \sum_{i=1}^{n_{dl}} v(\bar{a}_i) \phi_i(x).$$

#### 3.3.1 Convergencia

Por último, veamos un resultado de convergencia de este método. Para ello, comencemos presentando la definición de malla regular:

**Definición 3.12.** Sea  $(\mathcal{T}_h)_{h>0}$  una sucesión de mallas de  $\Omega$ . Se dice que es una sucesión de mallas regulares si

### 3. FUNDAMENTOS TEÓRICOS

---

1. la sucesión  $h = \max_{K_i \in \mathcal{T}_h} \text{diam}(K_i)$  tiende a 0.
2. existe una constante  $C$  tal que, para todo  $h > 0$  y todo  $K \in \mathcal{T}_h$ ,

$$\frac{\text{diam}(K)}{\rho(K)} \leq C,$$

donde  $\rho(K)$  denota el diámetro de la mayor bola contenida en  $K$ .

**Teorema 3.7.** Sea  $(\mathcal{T}_h)_{h>0}$  una sucesión de mallas regulares de  $\Omega$ . Sea  $u \in H_0^1(\Omega)$  la solución del problema de Dirichlet (3.3), y  $u_h \in V_{0h}$  su aproximación interna por elementos finitos  $\mathbb{P}_k$ . Entonces, el Método de los Elementos Finitos  $\mathbb{P}_k$  converge; en concreto,

$$\lim_{h \rightarrow 0} \|u - u_h\|_{H^1(\Omega)} = 0.$$

Más aún, si  $u \in H^{k+1}(\Omega)$  y si  $k + 1 > \frac{N}{2}$ , entonces tenemos la siguiente estimación de error de orden  $k$ :

$$\|u - u_h\|_{H^1(\Omega)} \leq Ch^k \|u\|_{H^{k+1}(\Omega)}$$

donde  $C$  es una constante independiente de  $h$  y de  $u$ .

Para la demostración de este teorema, ver, por ejemplo, [8, 10].

#### 3.4 El método de las líneas

Finalmente, como teoría necesaria para el posterior estudio de los métodos paralelos de resolución para EDP, presentamos el método de las líneas. Dicho método consiste en la discretización en espacio mediante elementos finitos de una EDP evolutiva, transformando el problema en un sistema de ecuaciones diferenciales ordinarias. Información más detallada puede encontrarse, por ejemplo, en [8] sección 8.6.

Tomaremos como modelo de EDP de evolución (de tipo parabólico) a la ecuación del calor:

$$\begin{aligned} u_t - \Delta u &= f, & \Omega \times (0, T), \\ u &= 0, & \partial\Omega \times (0, T), \\ u(x, 0) &= u_0(x), & x \in \Omega, \end{aligned} \tag{3.6}$$

donde  $\Omega \subset \mathbb{R}^d$ ,  $d = 1, 2, 3, \dots$  y  $T > 0$ .

Para la discretización de la ecuación del calor utilizaremos, primero, elementos finitos en espacio, antes de abordar la discretización en tiempo. Así, consideremos un espacio

$V_h \subset H_0^1(\Omega)$  formado por elementos finitos  $\mathbb{P}_k$  sobre una malla  $\mathcal{T}_h$ .

Multiplicando (3.6) por funciones test  $v_h \in V_h$  e integrando por partes, se llega al problema discreto en espacio: Hallar  $u_h(t) : (0, T) \rightarrow V_h$  tal que

$$\begin{cases} \int_{\Omega} \partial_t u_h(t) v_h + \int_{\Omega} \nabla u_h(t) \nabla v_h = \int_{\Omega} f(t) v_h, & \text{para todo } v_h \in V_h, \\ u_h(t=0) = u_{h,0}, \end{cases} \quad (3.7)$$

donde  $u_{h,0} \in V_h$  es una aproximación del dato inicial.

Vamos a ver que (3.7) se puede escribir como un problema de valor inicial para un sistema de  $N$  ecuaciones diferenciales ordinarias de orden 1. Para ello, escribamos  $u_h(t)$  como combinación lineal de una base  $\{\varphi_j\}_{j=1}^N$  de  $V_h$ :

$$u_h(t) = \sum_{j=1}^N u_j(t) \varphi_j \in V_h.$$

Donde  $\mathbf{u}(t) = (u_j(t))_{j=1}^N$  es un vector de  $\mathbb{R}^N$ , para cada  $t \in (0, T)$ . Es importante señalar que las funciones base no dependen del tiempo, solo de las coordenadas.

Entonces, (3.7) es equivalente al siguiente sistema de  $N$  ecuaciones:

$$\begin{cases} \sum_{j=1}^N \left( \int_{\Omega} \varphi_i \varphi_j \right) \partial_t u_j(t) + \sum_{j=1}^N \left( \int_{\Omega} \nabla \varphi_i \nabla \varphi_j \right) u_j(t) = \int_{\Omega} f(t) \varphi_i, & \text{para todo } i = 1, \dots, N, \\ u_j(t=0) = u_j^0, \end{cases} \quad (3.8)$$

donde los  $u_j^0$  (en el dato inicial) son las coordenadas de  $u_{h,0}$ ; es decir,  $u_{h,0} = \sum_{j=1}^N u_j^0 \varphi_j$ .

Véase que (3.8) es un sistema de EDO de primer orden que, de hecho, se puede escribir matricialmente como

$$\begin{cases} M \frac{d\mathbf{u}(t)}{dt} + K \mathbf{u}(t) = F(t), & t \in (0, T), \\ \mathbf{u}(t=0) = \mathbf{u}^0, \end{cases}$$

donde  $K$  es la matriz de rigidez y  $M$  es la denominada matriz de masa:

$$M_{i,j} = \int_{\Omega} \varphi_i \varphi_j, \quad K_{i,j} = \int_{\Omega} \nabla \varphi_i \nabla \varphi_j.$$



# Métodos de disparo

En este capítulo expondremos el método de disparo y el método de disparo múltiple. Se trata de dos procesos secuenciales que nos serán de utilidad a la hora de desarrollar el método paralelo en tiempo, objetivo de nuestro estudio.

## 4.1 Método de disparo

En primer lugar, vamos a comenzar explicando el método de disparo, en el cual se encuentra la idea que desarrollaremos en la siguiente sección para estudiar el método de disparo múltiple. Un estudio detallado puede encontrarse, por ejemplo, en [13].

El objetivo de este procedimiento es convertir un problema de contorno en uno de valores iniciales, más simple, que podamos resolver mediante algún método numérico. Para ello, consideremos:

$$\begin{aligned}\partial_t u(t) &= f(u(t)), & t \in (0, L), \\ u(0) &= g_0, \\ u(L) &= g_L\end{aligned}\tag{4.1}$$

Es claro que, para convertir este problema en uno de valores iniciales, necesitamos una condición inicial para la derivada parcial de  $u$  respecto al tiempo. Dicho valor se denomina *parámetro de disparo* y vamos a denotarlo por la letra  $Z$ .

#### 4. MÉTODOS DE DISPARO

---

Así, nuestro objetivo es transformar (4.1) en un problema de la forma:

$$\begin{aligned}\partial_t u(t) &= f(u(t)), & t \in (0, L), \\ u(0) &= g_0, \\ \partial_t(0) &= Z\end{aligned}\tag{4.2}$$

El propósito de este método es calcular el parámetro de disparo,  $Z$ , de modo que, para  $t = L$ , la solución alcance el valor  $g_L$  al menos de forma aproximada. El procedimiento de cálculo de  $Z$  es un proceso iterativo y consiste en, partiendo de un valor cualquiera de  $Z$ :

1. Si al resolver (4.2) el valor de la solución en  $t = L$  sobrepasa a  $g_L$ , tomamos un valor menor para  $Z$  y repetimos.
2. Análogamente, si  $g_L > u(L)$ , ajustamos el parámetro  $Z$  con un valor mayor y realizamos una nueva iteración.
3. Si al resolver el problema obtenemos que la solución verifica  $u(L) = g_L$ , hemos terminado debido a la unicidad de solución.

La resolución de (4.2) se lleva a cabo mediante algún método numérico (como Euler o Runge-Kutta). Presentamos ahora un ejemplo para su mayor comprensión:

**Ejemplo 4.1.** Consideremos el problema de contorno

$$\begin{aligned}\partial_t u(t) &= 3t^2 + 2t, & t \in (0, 4), \\ u(0) &= 1, \\ u(4) &= 12\end{aligned}\tag{4.3}$$

Siguiendo con el proceso expuesto anteriormente, nuestro objetivo es transformarlo en un problema de valores iniciales de la forma:

$$\begin{aligned}\partial_t u(t) &= 3t^2 + 2t, & t \in (0, 4), \\ u(0) &= 1, \\ \partial_t(0) &= Z\end{aligned}\tag{4.4}$$

Iteración 1. Para una primera iteración, tomemos el valor  $Z = 3$  y resolvamos (4.4) mediante el método de Euler. Obtenemos que  $u(4) \simeq 85,48 > 12$ .

Iteración 2. Procedemos a una segunda iteración con un valor menor que  $Z = 3$ , por ejemplo,  $Z = 0$ . Ahora obtenemos  $u(4) \simeq 53,48 > 12$ , luego tenemos que disminuir más aún el parámetro de disparo.

Iteración 3. Tomamos  $Z = -15$  y volvemos a resolver (4.4) mediante el método de Euler, obteniendo  $u(4) \simeq 25,48 > 12$ .

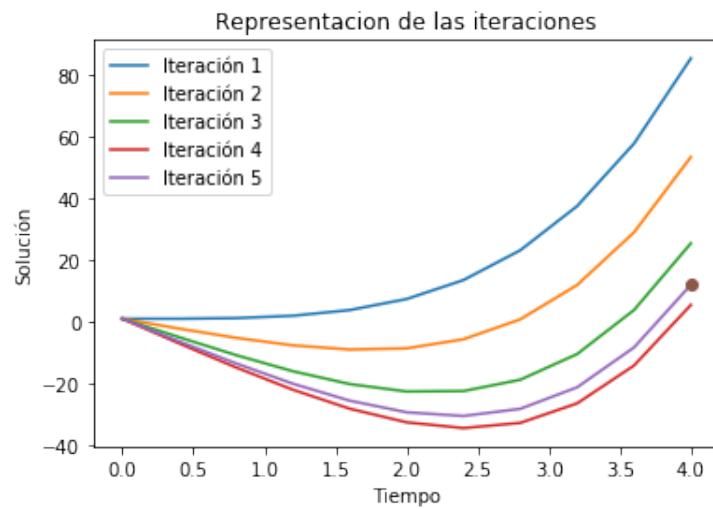


## 4.2 Método de disparo múltiple en tiempo

Iteración 4. Tomando ahora el valor  $Z = -20$ , tras resolver de nuevo mediante Euler, el resultado obtenido es  $u(4) \simeq 5,48 < 12$ . En esta iteración hemos obtenido un valor menor que el deseado, luego tenemos que incrementar  $Z$  para la siguiente iteración.

Iteración 5. Si tomamos  $Z = -18,37$  el valor de la solución en el instante final coincide con el deseado; es decir,  $u(4) \simeq 12,00$ .

Veamos la representación gráfica de estas iteraciones, donde el punto gris es el valor  $(4, 12)$ , por el cual debe pasar la solución:



Una vez entendida la idea de este método de resolución mediante "disparos", podemos formalizarlo: buscamos transformar el problema (4.1) en uno de la forma (4.2) de modo que se satisfaga la ecuación

$$F(Z) := u(L, Z) - g_L = 0.$$

## 4.2 Método de disparo múltiple en tiempo

Abandonamos ahora la idea de transformar un problema de contorno en uno de valores iniciales para centrarnos en la resolución de este último. Más información acerca de estos métodos puede encontrarse consultando, por ejemplo, [14, 15].

Recordando la propuesta innovadora de Nievergelt en 1964 [1], el primer paso será dividir nuestro problema en tantos subproblemas como subintervalos del dominio temporal queramos considerar. Es claro que, salvo las condiciones iniciales para el primer subproblema, las demás son desconocidas. Dichas condiciones son las que ahora denominaremos

#### 4. MÉTODOS DE DISPARO

---

*parámetros de disparo*, en similitud con el método anterior.

Pasaremos ahora a exponer el proceso, para lo cual haremos uso de la siguiente ecuación diferencial:

$$\begin{aligned}\partial_t \mathbf{u}(t) &= \mathbf{f}(t, \mathbf{u}(t)), & t \in (0, T], \\ \mathbf{u}(0) &= \mathbf{u}^0\end{aligned}\tag{4.5}$$

Descomponemos el dominio temporal en subintervalos de la forma  $(T_n, T_{n+1}]$ , para  $n = 0, \dots, N-1$ . De esta forma, obtenemos los subproblemas:

$$\begin{aligned}\partial_t \mathbf{u}_n(t) &= \mathbf{f}(t, \mathbf{u}_n(t)), & t \in (T_n, T_{n+1}], \\ \mathbf{u}_n(T_n) &= \mathbf{U}_n.\end{aligned}\tag{4.6}$$

La resolución de cada uno de estos problemas de valores iniciales requiere el conocimiento de los parámetros  $\mathbf{U}_n$ , a los que llamamos parámetros de disparo. Como dijimos antes, es claro que dichos valores son desconocidos, salvo  $\mathbf{U}_0$ , y ahora no poseemos condiciones de contorno que nos ayuden en la búsqueda de estos. Por ello, nos apoyaremos en la continuidad de la solución, de la cual obtenemos las siguientes igualdades.

$$\begin{aligned}\mathbf{U}_0 &= \mathbf{u}_0(0) = \mathbf{u}^0 \\ \mathbf{U}_1 &= \mathbf{u}_0(T_1) = \mathbf{u}_0(T_1, \mathbf{U}_0) \\ \mathbf{U}_2 &= \mathbf{u}_1(T_2) = \mathbf{u}_1(T_2, \mathbf{U}_1) \\ &\dots \\ \mathbf{U}_N &= \mathbf{u}_{N-1}(T_N) = \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}),\end{aligned}\tag{4.7}$$

donde por  $\mathbf{u}_n(\cdot) = \mathbf{u}_n(\cdot, \mathbf{U}_{n-1})$  ( $1 \leq n < N$ ) denotamos a la solución de (4.6).

Vemos así, como cabía esperar, que cada  $\mathbf{U}_n$  depende del instante inicial  $T_n$  y de la solución del subproblema anterior,  $\mathbf{u}_{n-1}$ , que está determinada por su valor inicial  $\mathbf{U}_{n-1}$ .

Si consideramos el vector  $\mathbf{U} := (\mathbf{U}_0, \mathbf{U}_1, \dots, \mathbf{U}_N)^T$ , nuestro objetivo se convierte en resolver el sistema:

$$F(\mathbf{U}) := \begin{pmatrix} \mathbf{U}_0 - \mathbf{u}^0 \\ \mathbf{U}_1 - \mathbf{u}_0(T_1, \mathbf{U}_0) \\ \mathbf{U}_2 - \mathbf{u}_1(T_2, \mathbf{U}_1) \\ \dots \\ \mathbf{U}_N - \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}) \end{pmatrix} = 0.\tag{4.8}$$

## 4.2 Método de disparo múltiple en tiempo

En la resolución de dicho sistema aplicaremos el método de Newton, por lo que la notación se nos vuelve algo compleja. Fijaremos: los subíndices denotan los subintervalos temporales, mientras que los superíndices serán las iteraciones del método de Newton.

Comenzaremos con una estimación inicial del vector  $\mathbf{U}^0$ , y a partir de ahí realizaremos las iteraciones:

$$\mathbf{U}^{k+1} = \mathbf{U}^k - [F'(\mathbf{U}^k)]^{-1} F(\mathbf{U}^k), \quad \text{para } k = 0, 1, 2, \dots$$

Para ello, en primer lugar, multiplicamos a ambos lados de la igualdad por  $F'(\mathbf{U}^k)$ , obteniendo:

$$[F'(\mathbf{U}^k)]\mathbf{U}^{k+1} = [F'(\mathbf{U}^k)] \cdot \mathbf{U}^k - F(\mathbf{U}^k) \quad (4.9)$$

Donde la matriz jacobiana  $F'(\mathbf{U}^k)$  es:

$$F'(\mathbf{U}^k) = \begin{bmatrix} I & & & & \\ -\frac{\partial}{\partial \mathbf{U}_0} \mathbf{u}_0(T_1, \mathbf{U}_0^k) & I & & & \\ & -\frac{\partial}{\partial \mathbf{U}_1} \mathbf{u}_1(T_2, \mathbf{U}_1^k) & & I & \\ & & \ddots & & \\ & & & -\frac{\partial}{\partial \mathbf{U}_{N-1}} \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) & I \end{bmatrix}$$

Así, el primer miembro de la igualdad (4.9) queda como:

$$\begin{pmatrix} \mathbf{U}_0^{k+1} \\ -\frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^{k+1} + \mathbf{U}_1^{k+1} \\ \vdots \\ -\frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^{k+1} + \mathbf{U}_N^{k+1} \end{pmatrix}$$

y el segundo:

$$\begin{aligned} & \begin{pmatrix} \mathbf{U}_0^k \\ -\frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^k + \mathbf{U}_1^k \\ \vdots \\ -\frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^k + \mathbf{U}_N^k \end{pmatrix} - \begin{pmatrix} \mathbf{U}_0^k - \mathbf{u}^0 \\ \mathbf{U}_1^k - \mathbf{u}_0(T_1, \mathbf{U}_0^k) \\ \vdots \\ \mathbf{U}_N^k - \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) \end{pmatrix} = \\ & = \begin{pmatrix} \mathbf{u}^0 \\ -\frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^k + \mathbf{u}_0(T_1, \mathbf{U}_0^k) \\ \vdots \\ -\frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^k + \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) \end{pmatrix} \end{aligned}$$

#### 4. MÉTODOS DE DISPARO

---

Luego, (4.9) nos queda de la forma:

$$\begin{pmatrix} \mathbf{U}_0^{k+1} \\ -\frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^{k+1} + \mathbf{U}_1^{k+1} \\ \vdots \\ -\frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^{k+1} + \mathbf{U}_N^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^0 \\ -\frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^k + \mathbf{u}_0(T_1, \mathbf{U}_0^k) \\ \vdots \\ -\frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^k + \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) \end{pmatrix}$$

Ahora, despejando el vector  $\mathbf{U}^{k+1}$ , obtenemos el sistema:

$$\begin{aligned} \mathbf{U}_0^{k+1} &= \mathbf{u}^0 \\ \mathbf{U}_1^{k+1} &= \frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^{k+1} - \frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot \mathbf{U}_0^k + \mathbf{u}_0(T_1, \mathbf{U}_0^k) \\ &\vdots \\ \mathbf{U}_N^{k+1} &= \frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^{k+1} - \frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k) \cdot \mathbf{U}_{N-1}^k + \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) \end{aligned}$$

Por último, agrupando términos:

$$\begin{aligned} \mathbf{U}_0^{k+1} &= \mathbf{u}^0 \\ \mathbf{U}_1^{k+1} &= \mathbf{u}_0(T_1, \mathbf{U}_0^k) + \frac{\partial \mathbf{u}_0}{\partial \mathbf{U}_0}(T_1, \mathbf{U}_0^k) \cdot (\mathbf{U}_0^{k+1} - \mathbf{U}_0^k) \\ &\vdots \\ \mathbf{U}_N^{k+1} &= \mathbf{u}_{N-1}(T_N, \mathbf{U}_{N-1}^k) + \frac{\partial \mathbf{u}_{N-1}}{\partial \mathbf{U}_{N-1}}(T_N, \mathbf{U}_{N-1}^k)(\mathbf{U}_{N-1}^{k+1} - \mathbf{U}_{N-1}^k) \end{aligned} \tag{4.10}$$

A la hora de implementar este método, observamos que también es necesaria una expresión para las derivadas  $\frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(t, \mathbf{U}_n)$ . Haciendo uso del problema inicial (4.6), cuya expresión es  $\partial_t \mathbf{u}_n(t) = \mathbf{f}(t, \mathbf{u}_n(t))$ , y denotando  $V_n(t) := \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(t, \mathbf{U}_n)$ , tenemos que:

$$\begin{aligned} \frac{\partial}{\partial t} V_n(t) &= \frac{\partial}{\partial t} \left[ \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(t, \mathbf{U}_n) \right] = \frac{\partial}{\partial \mathbf{U}_n} \left[ \frac{\partial}{\partial t} \mathbf{u}_n(t, \mathbf{U}_n) \right] = \frac{\partial}{\partial \mathbf{U}_n} [\mathbf{f}(t, \mathbf{u}_n(t, \mathbf{U}_n))] = \\ &= \frac{\partial}{\partial \mathbf{u}_n} \mathbf{f}(t, \mathbf{u}_n(t, \mathbf{U}_n)) \cdot \frac{\partial}{\partial \mathbf{U}_n} \mathbf{u}_n(t, \mathbf{U}_n) = \frac{\partial}{\partial \mathbf{u}_n} \mathbf{f}(t, \mathbf{u}_n(t, \mathbf{U}_n)) \cdot V_n(t) \end{aligned}$$

**Observación 4.1.** Para la condición inicial tengamos en cuenta que, por la definición de los subproblemas y los parámetros de disparo,  $\mathbf{u}_n(T_n, \mathbf{U}_n) = \mathbf{U}_n$ . Por lo que

$$V_n(T_n) = \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_n, \mathbf{U}_n) = \frac{\partial}{\partial \mathbf{U}_n} \mathbf{U}_n = I.$$

Luego, podemos obtener los valores de  $V_n(t) = \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(t, \mathbf{U}_n^k)$  mediante la resolución de los sistemas:

$$\begin{cases} \partial_t V_n(t) &= \frac{\partial}{\partial \mathbf{U}_n} \mathbf{f}(t, \mathbf{u}_n(t, \mathbf{U}_n^k)) \cdot V_n(t) \\ V_n(T_n) &= I \end{cases} \quad t \in (T_n, T_{n+1})$$

**Observación 4.2.** Nótese que estos sistemas deben ser resueltos a la vez que (4.10), pues su expresión depende de las soluciones  $\mathbf{u}_n(t, \mathbf{U}_n^k)$ .

### Resultados sobre convergencia

Antes de pasar a nuestro objetivo final, la paralelización, veamos algunos resultados sobre la eficiencia de lo que acabamos de estudiar.

**Teorema 4.1** (Convergencia cuadrática). Si la función  $f(t, \mathbf{u}(t))$  del problema de valores iniciales (4.5) es de clase  $\mathcal{C}^2$  respecto a su segundo argumento, entonces el método de disparo múltiple aplicado a su resolución converge localmente cuadráticamente; es decir,

$$\|\mathbf{U} - \mathbf{U}^{k+1}\| \leq \frac{1}{2} \| [F'(\mathbf{U}^k)]^{-1} \| \cdot \| F''(\mathbf{U}^k) \| \cdot \|\mathbf{U} - \mathbf{U}^k\|^2 + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3),$$

donde  $F'(\mathbf{U})$  denota la matriz jacobiana de  $F(\mathbf{U})$  y  $F''(\mathbf{U})$  al operador bilineal que representa la segunda derivada de  $F(\mathbf{U})$ .

*Demostración.* En primer lugar, aplicando la hipótesis de diferenciabilidad de  $f$  vamos a desarrollar  $F(\mathbf{U})$  en la iteración  $k$ -ésima:

$$F(\mathbf{U}) = F(\mathbf{U}^k) + F'(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k) + \frac{1}{2} F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3) \quad (4.11)$$

Ahora, aplicando el método de Newton a  $F(\mathbf{U})$  y operando:

$$\begin{aligned} \mathbf{U}^{k+1} &= \mathbf{U}^k - [F'(\mathbf{U}^k)]^{-1} F(\mathbf{U}^k), \\ F'(\mathbf{U}^k) \mathbf{U}^{k+1} &= F'(\mathbf{U}^k) \mathbf{U}^k - F(\mathbf{U}^k), \\ 0 &= F'(\mathbf{U}^k)(\mathbf{U}^k - \mathbf{U}^{k+1}) - F(\mathbf{U}^k), \\ F(\mathbf{U}^k) &= -F'(\mathbf{U}^k)(\mathbf{U}^{k+1} - \mathbf{U}^k). \end{aligned}$$

Si  $\mathbf{U}$  es la solución al sistema, entonces  $F(\mathbf{U}) = 0$  y, sustituyendo en (4.11), tenemos:

$$0 = F(\mathbf{U}^k) + F'(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k) + \frac{1}{2} F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3).$$

#### 4. MÉTODOS DE DISPARO

---

Sustituyendo aquí la última expresión obtenida para  $F(\mathbf{U}^k)$ :

$$0 = -F'(\mathbf{U}^k)(\mathbf{U}^{k+1} - \mathbf{U}^k) + F'(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k) + \frac{1}{2}F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3)$$

$$0 = F'(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^{k+1}) + \frac{1}{2}F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3).$$

Despejamos  $(\mathbf{U} - \mathbf{U}^{k+1})$ :

$$-F'(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^{k+1}) = \frac{1}{2}F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3),$$

$$-(\mathbf{U} - \mathbf{U}^{k+1}) = \frac{1}{2}[F'(\mathbf{U}^k)]^{-1}F''(\mathbf{U}^k)(\mathbf{U} - \mathbf{U}^k, \mathbf{U} - \mathbf{U}^k) + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3).$$

Por último, tomando normas y aplicando la desigualdad triangular en el miembro de la derecha:

$$\|\mathbf{U} - \mathbf{U}^{k+1}\| \leq \frac{1}{2}\|[F'(\mathbf{U}^k)]^{-1}\| \cdot \|F''(\mathbf{U}^k)\| \cdot \|\mathbf{U} - \mathbf{U}^k\|^2 + \mathcal{O}(\|\mathbf{U} - \mathbf{U}^k\|^3),$$

Obteniendo así la expresión que buscábamos. □

**Teorema 4.2** (Convergencia en paso finito). Si  $\mathbf{u}^0 = \mathbf{u}^0$ , entonces el método de disparo múltiple tiene la siguiente propiedad:

$$\mathbf{U}_n^k = \mathbf{u}(T_n), \quad \text{si } k \geq n.$$

Es decir,  $\mathbf{U}_n^k$  coincide con la solución exacta desde la iteración  $k = n$  en adelante.

*Demostración.* Realizaremos la demostración de este resultado por inducción en la dirección del tiempo.

Sea  $\mathbf{n} = 0$ , por hipótesis tenemos que  $\mathbf{u}^0 = \mathbf{u}^0$  y  $\mathbf{u}^{k+1} = \mathbf{u}^0$  para  $k = 0, 1, 2, \dots$  como consideramos en (4.10). Luego, el valor de este parámetro es siempre exacto. Consideremos entonces la hipótesis cierta para  $n$ ; es decir,  $\mathbf{U}_n^k = \mathbf{u}(T_n)$  para  $k \geq n$ .

Sea  $\mathbf{n} + 1$ . De (4.10) tenemos que

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_{n+1}, \mathbf{U}_n^k)(\mathbf{U}_n^{k+1} - \mathbf{U}_n^k).$$

Para probar el resultado, tenemos que tomar  $k + 1 \geq n + 1$ , lo que implica que  $k \geq n$  y por tanto  $\mathbf{U}_n^k = \mathbf{u}(T_n)$  y  $\mathbf{U}_n^{k+1} = \mathbf{u}(T_n)$ . Sustituyendo esto en la igualdad anterior:

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \mathbf{u}(T_n)) + \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_{n+1}, \mathbf{u}(T_n))(\mathbf{u}(T_n) - \mathbf{u}(T_n))$$

Simplificando:

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \mathbf{u}(T_n)).$$

Es claro que, por la definición que dimos a los parámetros de disparo, se verifica lo siguiente

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1})$$

Pero para concluir con el resultado que buscamos, necesitamos que el miembro de la derecha de la última igualdad verifique  $\mathbf{u}_n(T_{n+1}) = \mathbf{u}(T_{n+1})$ ; es decir,  $\mathbf{u}_n(T_{n+1}, \mathbf{u}(T_n)) = \mathbf{u}_n(T_{n+1})$ . lo que nos lleva a hacernos la siguiente pregunta: Si partimos del valor exacto para el parámetro de disparo en un subproblema, ¿obtenemos la solución exacta?

Si recordamos la definición de dichos subproblemas e imponemos la condición inicial a la que refiere  $\mathbf{u}_n(T_{n+1}, \mathbf{u}(T_n))$ , tenemos

$$\begin{cases} \partial_t \mathbf{u}_n(t) &= f(t, \mathbf{u}_n(t)), & t \in (T_n, T_{n+1}], \\ \mathbf{u}_n(T_n) &= \mathbf{u}(T_n) \end{cases}$$

Por la unicidad de solución, al resolver este problema que parte del dato inicial exacto y verifica la misma ecuación que el problema original, obtendremos la solución de dicho problema original, restringida al subintervalo en el que nos encontramos. Por lo tanto, la solución en el extremo final,  $\mathbf{u}_n(T_{n+1})$ , coincidirá con la exacta  $\mathbf{u}(T_{n+1})$ . Luego

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}(T_{n+1}) \quad \text{para } k + 1 \geq n + 1,$$

verificándose así la hipótesis de inducción. □





*On propose dans cette Note un schéma permettant de profiter d'une architecture parallèle pour la discretisation en temps d'une équation d'évolution aux dérivées partielles. [...] Elle a pour principale motivation les problèmes en temps réel, d'où la terminologie proposée de «pararéel».*

J.-L. Lions, Y. Maday, G. Turinici

CAPÍTULO

# 5

## Parareal

En esta sección nos centramos en el método clave de nuestro estudio. El algoritmo Parareal se introdujo por primera vez en 2001, de la mano de J.-L. Lions, Y. Maday y G. Turinici [3]. Se trata de un método de resolución paralelo en tiempo, mediante el cual podremos resolver ecuaciones en derivadas parciales. Tras su estudio teórico, pondremos a prueba el método resolviendo la ecuación del calor. Un desarrollo en profundidad y más ejemplos de este método pueden encontrarse en [16, 17].

La idea de este método se basa en la realización de dos etapas: una primera secuencial e imprecisa, y una segunda paralela y rigurosa. Así, la parte secuencial se realiza con métodos de aproximación rápidos pero poco precisos, para luego corregirse con soluciones minuciosas obtenidas en paralelo, ahorrando de esta forma grandes tiempos de cómputo.

Comenzaremos considerando el siguiente problema de ecuaciones en derivadas parciales:

$$\begin{aligned} u_t - \Delta u &= f(t, \mathbf{x}, u(t, \mathbf{x})), & (t, \mathbf{x}) &\in (0, T] \times \Omega, \\ u(\mathbf{x}, 0) &= u_0(\mathbf{x}), & \mathbf{x} &\in \Omega, \\ u(\mathbf{x}, t) &= g_0(t), & (t, \mathbf{x}) &\in (0, T] \times \partial\Omega. \end{aligned} \tag{5.1}$$

Por el método de las líneas (sección 3.4), sabemos que tras la discretización en espacio por elementos finitos, es posible transformar dicho problema en un sistema de ecuaciones diferenciales ordinarias. Por ello, centraremos la teoría que prosigue en el siguiente

## 5. PARAREAL

---

problema:

$$\begin{aligned}\partial_t \mathbf{u}(t) &= \mathbf{f}(t, \mathbf{u}(t)), \quad t \in (0, T], \\ \mathbf{u}(0) &= \mathbf{u}^0,\end{aligned}\tag{5.2}$$

donde  $\mathbf{u}$  es el vector cuyas coordenadas contienen a los grados de libertad del espacio de elementos finitos.

**Observación 5.1.** Por supuesto, esta teoría puede aplicarse al estudio de ecuaciones diferenciales ordinarias; no es necesario que las ecuaciones de (5.2) provengan de la discretización de una EDP. De hecho, Parareal tuvo su origen en la aplicación a ecuaciones diferenciales, tratándose en este trabajo una extensión del algoritmo a EDP.

Como en la sección anterior, retomamos la idea de Nievelgert y subdividimos el intervalo temporal en subintervalos:  $(T_n, T_{n+1}]$ , para  $n = 0, 1, \dots, N-1$ , donde  $T_0 = 0$  y  $T_N = T$ . Obtenemos los subproblemas:

$$\begin{aligned}\partial_t \mathbf{u}_n(t) &= \mathbf{f}(t, \mathbf{u}_n(t)), \quad t \in (T_n, T_{n+1}], \\ \mathbf{u}_n(T_n) &= \mathbf{U}_n,\end{aligned}\tag{5.3}$$

Nuestros parámetros de disparo son ahora los  $\mathbf{U}_n$ . La resolución de estos subproblemas se basará en dos operadores:

- $\mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n)$ : una aproximación poco precisa de la solución  $\mathbf{u}_n(T_{n+1})$  de (5.3), partiendo de la condición inicial  $\mathbf{u}_n(T_n) = \mathbf{U}_n$ .
- $\mathbf{F}(T_{n+1}, T_n, \mathbf{U}_n)$ : una aproximación más precisa de la solución  $\mathbf{u}_n(T_{n+1})$  de (5.3), partiendo de la condición inicial  $\mathbf{u}_n(T_n) = \mathbf{U}_n$ .

Ahora, para comenzar el proceso iterativo de resolución, partiremos de unos valores obtenidos mediante el operador de aproximación grosero,  $\mathbf{G}$ :

$$\mathbf{U}_0^0 := \mathbf{u}^0, \quad \mathbf{U}_{n+1}^0 := \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^0)$$

Y, para cada iteración, calculamos:

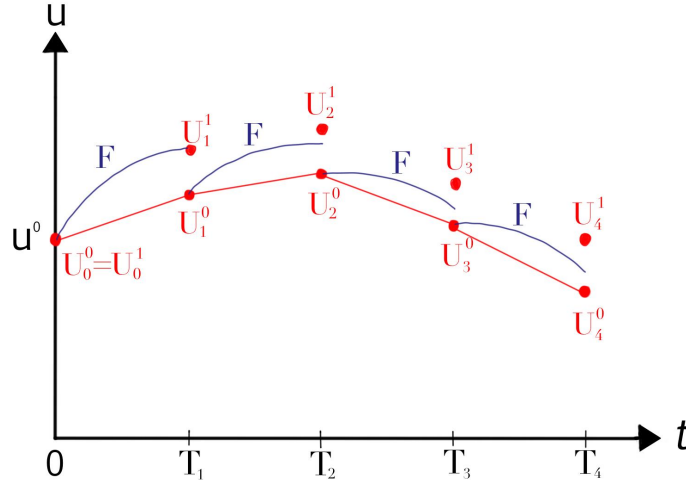
$$\begin{aligned}\mathbf{U}_0^{k+1} &:= \mathbf{u}^0, \\ \mathbf{U}_{n+1}^{k+1} &:= \mathbf{F}(T_{n+1}, T_n, \mathbf{U}_n^k) + \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^{k+1}) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^k).\end{aligned}\tag{5.4}$$

Como podemos observar, el operador preciso  $\mathbf{F}$  no requiere de parámetros calculados en la iteración actual, sino de los de la etapa anterior. Por este motivo, para todo  $n \in \{0, 1, \dots, N-1\}$ , podemos calcular en paralelo los sumandos  $\mathbf{F}(T_{n+1}, T_n, \mathbf{U}_n^k)$ , donde

reside el grueso de la computación.

Así, mediante el uso de este algoritmo, somos capaces de reducir el tiempo necesario para la resolución de un problema. Es claro que dicha mejora dependerá fuertemente de los núcleos de los que se disponga, así como de la cantidad de subintervalos en los que se divida el dominio temporal.

Con el fin de hacer más intuitiva la comprensión del procedimiento expuesto, añadimos la siguiente imagen:



En primer lugar, partiendo del valor inicial  $u^0$ , se calculan de manera secuencial los valores de  $U_0^0, U_1^0, U_2^0, U_3^0$  y  $U_4^0$ . A continuación, en paralelo, se calculan las aproximaciones precisas  $F$  partiendo de cada uno de los valores anteriores  $U_i^0$ . El último paso, del que se obtendrán los  $U_i^1$ , consiste en la corrección dada por la expresión (5.4), de nuevo realizada de manera secuencial.

## 5.1 Resultados teóricos

En esta sección vamos a exponer tres resultados sobre el método Parareal que nos motivarán a su aplicación práctica. El primero de ellos relaciona este algoritmo con lo visto en el capítulo anterior, los métodos de disparo múltiple, y con ello, también lo relaciona con sus resultados. Los dos teoremas siguientes hablan sobre la convergencia y la estimación del error, comenzando por una deducción semejante a la del capítulo anterior, y acabando con una generalización sobre el error del algoritmo Parareal.

## 5. PARAREAL

---

**Teorema 5.1.** El algoritmo Parareal es un método de disparo múltiple en el que se realiza una aproximación en los términos de la derivada (provenientes del Jacobiano en el método de Newton) mediante una diferencia de las trayectorias evaluadas en la malla gruesa.

*Demostración.* Comencemos recordando la estructura del método de disparo múltiple estudiado en la sección anterior:

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_{n+1}, \mathbf{U}_n^k)(\mathbf{U}_n^{k+1} - \mathbf{U}_n^k). \quad (5.5)$$

Por otra parte, utilizando el desarrollo en series de Taylor para  $\mathbf{u}_n(T_{n+1}, \mathbf{U}_n^{k+1})$ :

$$\mathbf{u}_n(T_{n+1}, \mathbf{U}_n^{k+1}) = \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_{n+1}, \mathbf{U}_n^k)(\mathbf{U}_n^{k+1} - \mathbf{U}_n^k) + \mathcal{O}(\|\mathbf{U}_n^{k+1} - \mathbf{U}_n^k\|^2),$$

obteniendo así la expresión de (5.5), más un término de orden 2 de la diferencia entre la solución en dos iteraciones consecutivas. Reordenando la igualdad:

$$\frac{\partial \mathbf{u}_n}{\partial \mathbf{U}_n}(T_{n+1}, \mathbf{U}_n^k)(\mathbf{U}_n^{k+1} - \mathbf{U}_n^k) = \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^{k+1}) - \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \mathcal{O}(\|\mathbf{U}_n^{k+1} - \mathbf{U}_n^k\|^2).$$

Sustituyendo ahora en (5.5):

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^{k+1}) - \mathbf{u}_n(T_{n+1}, \mathbf{U}_n^k) + \mathcal{O}(\|\mathbf{U}_n^{k+1} - \mathbf{U}_n^k\|^2).$$

Podemos definir un método de disparo múltiple aproximado, al que denotaremos con una tilde, eliminando el término  $\mathcal{O}(\|\mathbf{U}_n^{k+1} - \mathbf{U}_n^k\|^2)$ :

$$\tilde{\mathbf{U}}_{n+1}^{k+1} = \mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^k) + \mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^{k+1}) - \mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^k)$$

Y ahora, en lugar de cancelar el primer y último término del segundo miembro de la igualdad anterior, introduciremos distintas aproximaciones para cada uno de ellos. En concreto, consideremos una aproximación precisa para el primer término:

$$\mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^k) \rightarrow \mathbf{F}(t_{n+1}, T_n, \tilde{\mathbf{U}}_n^k).$$

Y, tomemos dos aproximaciones groseras para los otros dos términos:

$$\mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^{k+1}) \rightarrow \mathbf{G}(T_{n+1}, T_n, \tilde{\mathbf{U}}_n^{k+1}),$$

$$\mathbf{u}_n(T_{n+1}, \tilde{\mathbf{U}}_n^k) \rightarrow \mathbf{G}(T_{n+1}, T_n, \tilde{\mathbf{U}}_n^k).$$

Obtenemos:

$$\tilde{\mathbf{U}}_{n+1}^{k+1} = \mathbf{F}(t_{n+1}, T_n, \tilde{\mathbf{U}}_n^k) + \mathbf{G}(T_{n+1}, T_n, \tilde{\mathbf{U}}_n^{k+1}) - \mathbf{G}(T_{n+1}, T_n, \tilde{\mathbf{U}}_n^k).$$

□

El siguiente resultado muestra una adaptación del teorema 4.2, de convergencia en paso finito, donde en lugar de obtener la solución exacta de una etapa en adelante, obtenemos el valor de la aproximación precisa. La demostración es análoga a la de dicho teorema, por tanto, se omitirá en este caso.

**Teorema 5.2.** Sea  $\mathbf{U}_0^0 = \mathbf{u}^0$ , entonces, para el algoritmo Parareal (5.4), tenemos la siguiente propiedad

$$\mathbf{U}_n^k = \mathbf{F}(T_n, 0, \mathbf{u}^0), \text{ si } k \geq n,$$

es decir,  $\mathbf{U}_n^k$  coincide con la aproximación fina desde la iteración  $k = n$  en adelante.

Por último, presentamos un resultado más general, que nos permitirá tener una cota para el error cometido dependiendo de cuantas iteraciones del método realicemos. Como  $\mathbf{F}$  es la aproximación precisa y podremos aumentar su precisión tanto como se desee, pues es el cálculo que realizaremos de manera paralela, por simplicidad, consideraremos que su solución es la exacta.

**Teorema 5.3** (Estimación de Convergencia de Parareal). Consideremos subintervalos del dominio temporal uniformes de longitud  $\Delta T$ . Sea  $\mathbf{F}$  la solución exacta de (5.1) y sea  $\mathbf{G}$  una solución aproximada con error local de truncamiento acotado por  $C_3 \Delta T^{p+1}$  y tal que satisface la condición de Lipschitz

$$\|\mathbf{G}(t + \Delta T, t, \mathbf{v}) - \mathbf{G}(t + \Delta T, t, \mathbf{w})\| \leq (1 + C_2 \Delta T) \|\mathbf{v} - \mathbf{w}\|. \quad (5.6)$$

Si la diferencia entre la solución aproximada dada por  $\mathbf{G}$  y la solución exacta representada por  $\mathbf{F}$  puede desarrollarse, para  $\Delta T$  pequeño, de la forma

$$\mathbf{F}(T_n, T_{n-1}, x) - \mathbf{G}(T_n, T_{n-1}, x) = c_{p+1}(x) \Delta T^{p+1} + c_{p+2}(x) \Delta T^{p+2} + \dots \quad (5.7)$$

donde los  $c_j$  son funciones de clase  $\mathcal{C}^1$  para  $j \in \{p+1, p+2, \dots\}$ , entonces, en la iteración  $k$  del algoritmo Parareal, tenemos la cota

$$\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| \leq \frac{C_3}{C_1} \frac{(C_1 \Delta T^{p+1})^{k+1}}{(k+1)!} (1 + C_2 \Delta T)^{n-k-1} \prod_{l=0}^k (n-l) \quad (5.8)$$

$$\leq \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2(T_n - T_{k+1})} \Delta T^{p(k+1)} \quad (5.9)$$

donde  $C_1$  es una constante que depende de los  $c_j$ .

*Demostración.* En primer lugar, como  $\mathbf{F}$  es la solución exacta del problema, tenemos que se verifica

$$\mathbf{u}(T_{n+1}) = \mathbf{F}(T_{n+1}, T_n, \mathbf{u}(T_n))$$

## 5. PARAREAL

---

es decir, el valor obtenido por  $\mathbf{F}$  partiendo de la solución exacta en el subintervalo anterior, es la solución exacta en este subintervalo.

Ahora, por la definición de los  $\mathbf{U}_{n+1}^{k+1}$  dada en el algoritmo Parareal (5.4), tenemos

$$\mathbf{U}_{n+1}^{k+1} = \mathbf{F}(T_{n+1}, T_n, \mathbf{U}_n^k) + \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^{k+1}) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^k).$$

De ambas expresiones, sumando y restando  $\mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n))$  y reordenando los términos,

$$\begin{aligned} \mathbf{u}(T_{n+1}) - \mathbf{U}_{n+1}^{k+1} &= \mathbf{F}(T_{n+1}, T_n, \mathbf{u}(T_n)) - \mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n)) \\ &\quad - (\mathbf{F}(T_{n+1}, T_n, \mathbf{U}_n^k) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^k)) \\ &\quad + \mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n)) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^{k+1}) \end{aligned}$$

Por (5.7),

$$\begin{aligned} \mathbf{u}(T_{n+1}) - \mathbf{U}_{n+1}^{k+1} &= c_{p+1}(\mathbf{u}(T_n))\Delta T^{p+1} + c_{p+2}(\mathbf{u}(T_n))\Delta T^{p+2} + \dots \\ &\quad - (c_{p+1}(\mathbf{U}_n^k)\Delta T^{p+1} + c_{p+2}(\mathbf{U}_n^k)\Delta T^{p+2} + \dots) \\ &\quad + \mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n)) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^{k+1}) \end{aligned} \quad (5.10)$$

Como los  $c_j$  son de clase  $\mathcal{C}^1$ , sabemos que existe una constante,  $C_1$ , tal que

$$\begin{aligned} \|(c_{p+1}(\mathbf{u}(T_n)) - c_{p+1}(\mathbf{U}_n^k))\Delta T^{p+1} + (c_{p+2}(\mathbf{u}(T_n)) - c_{p+2}(\mathbf{U}_n^k))\Delta T^{p+2} + \dots\| &\leq \\ &\leq C_1\Delta T^{p+1}\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| \end{aligned}$$

Tomando normas en (5.10) y usando la hipótesis de la condición de Lipschitz para  $\mathbf{G}$ ,

$$\|\mathbf{u}(T_{n+1}) - \mathbf{U}_{n+1}^{k+1}\| \leq C_1\Delta T^{p+1}\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| + (1 + C_2\Delta T)\|\mathbf{u}(T_n) - \mathbf{U}_n^{k+1}\|.$$

Para acotar este error, vamos a estudiar la relación de recurrencia con igualdad:

$$e_{n+1}^{k+1} = \alpha e_n^k + \beta e_n^{k+1}, \quad e_{n+1}^0 = \gamma + \beta e_n^0 \quad (5.11)$$

donde  $\alpha := C_1\Delta T^{p+1}$ ,  $\beta := 1 + C_2\Delta T$ ,  $\gamma := C_3\Delta T^{p+1}$  y la inicialización para (5.11) se toma de la inicialización del algoritmo Parareal siguiendo el mismo procedimiento que antes:

$$\begin{aligned} \|\mathbf{u}(T_{n+1}) - \mathbf{U}_{n+1}^0\| &= \|\mathbf{u}(T_{n+1}) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^0)\| \\ &= \|\mathbf{u}(T_{n+1}) - \mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n)) + \mathbf{G}(T_{n+1}, T_n, \mathbf{u}(T_n)) - \mathbf{G}(T_{n+1}, T_n, \mathbf{U}_n^0)\| \\ &\leq C_3\Delta T^{p+1} + (1 + C_2\Delta T)\|\mathbf{u}(T_n) - \mathbf{U}_n^0\|. \end{aligned}$$

Ahora, para estudiar la relación de recurrencia, comenzaremos multiplicando la expresión por  $\xi^{n+1}$  y sumando en  $n$ ,

$$\sum_{n=0}^{\infty} e_{n+1}^{k+1} \xi^{n+1} = \alpha \sum_{n=0}^{\infty} e_n^k \xi^{n+1} + \beta \sum_{n=0}^{\infty} e_n^{k+1} \xi^{n+1} = \alpha \xi \sum_{n=0}^{\infty} e_n^k \xi^n + \beta \xi \sum_{n=0}^{\infty} e_n^{k+1} \xi^n$$

Como  $e_0^k = \|\mathbf{u}(0) - \mathbf{U}_0^k\| = 0$ , podemos empezar los sumatorios de la derecha de la igualdad en  $n = 1$ ,

$$\sum_{n=1}^{\infty} e_n^{k+1} \xi^n = \alpha \xi \sum_{n=1}^{\infty} e_n^k \xi^n + \beta \xi \sum_{n=1}^{\infty} e_n^{k+1} \xi^n.$$

Luego, la función generadora  $\rho_k(\xi) := \sum_{n=1}^{\infty} e_n^k \xi^n$ , satisface la relación de recurrencia:  $\rho_{k+1}(\xi) = \alpha \xi \rho_k(\xi) + \beta \xi \rho_{k+1}(\xi)$ . Tenemos entonces

$$\rho_{k+1}(\xi) = \frac{\alpha \xi}{1 - \beta \xi} \rho_k(\xi) = \frac{\alpha^{k+1} \xi^{k+1}}{(1 - \beta \xi)^{k+1}} \rho_0(\xi). \quad (5.12)$$

Ahora, para calcular  $\rho_0(\xi)$ , partimos de la inicialización en (5.11) y procedemos como antes,

$$\begin{aligned} \sum_{n=0}^{\infty} e_{n+1}^0 \xi^{n+1} &= \gamma \xi \sum_{n=0}^{\infty} \xi^n + \beta \xi \sum_{n=0}^{\infty} e_n^0 \xi^n, \\ \rho_0(\xi) &= \gamma \frac{\xi}{1 - \xi} + \beta \xi \rho_0(\xi) \implies \rho_0(\xi) = \frac{\gamma \xi}{(1 - \xi)(1 - \beta \xi)}. \end{aligned}$$

Sustituyendo en (5.12) y cambiando  $k + 1$  por  $k$ ,

$$\rho_k(\xi) = \gamma \alpha^k \frac{\xi^{k+1}}{(1 - \xi)(1 - \beta \xi)^{k+1}}.$$

Procedemos ahora a calcular los coeficientes de esta expresión, pues son los que acotan el error de Parareal. Para ello, en primer lugar, nos conviene sustituir  $(1 - \xi)$  por  $(1 - \beta \xi)$ , lo cual, dado que  $\beta \geq 1$ , solo afectará en que la cota obtenida sea mayor. Así, consideremos la función generatriz modificada:

$$\tilde{\rho}_k(\xi) = \gamma \alpha^k \frac{\xi^{k+1}}{(1 - \beta \xi)^{k+2}}.$$

Consideramos la fórmula de la serie binomial general:

$$\frac{1}{(1 - z)^{b+1}} = \sum_{j=0}^{\infty} \binom{b+j}{j} z^j,$$

que en nuestro caso sería

$$\frac{1}{(1 - \beta \xi)^{k+2}} = \sum_{j=0}^{\infty} \binom{k+1+j}{j} (\beta \xi)^j.$$

## 5. PARAREAL

---

$$\tilde{\rho}_k(\xi) = \gamma \alpha^k \sum_{j=0}^{\infty} \binom{k+1+j}{j} \beta^j \xi^{j+k+1} = \gamma \alpha^k \sum_{n=k+1}^{\infty} \binom{n}{n-k-1} \beta^{n-k-1} \xi^n.$$

Así, vemos que los coeficientes para  $n \leq k$  son nulos, como obteníamos también en el teorema 5.2. Ahora, usando que

$$\binom{n}{n-k-1} = \binom{n}{k+1} = \frac{1}{(k+1)!} \prod_{l=0}^k (n-l),$$

$$\tilde{\rho}_k(\xi) = \frac{\gamma \alpha^k}{(k+1)!} \sum_{n=k+1}^{\infty} \prod_{l=0}^k (n-l) \beta^{n-k-1} \xi^n,$$

tenemos que el coeficiente  $n$ -ésimo  $e_n^k$  satisface, para  $n > k$ :

$$e_n^k \leq \frac{\gamma \alpha^k}{(k+1)!} \beta^{n-k-1} \prod_{l=0}^k (n-l),$$

y observamos que esta cota también tiene valor nulo para  $n \leq k$ , pues el productorio proveniente del coeficiente binomial desaparece para los  $k > n$ .

Finalmente, aplicando lo obtenido y sustituyendo los valores de  $\alpha$ ,  $\beta$  y  $\gamma$ , obtenemos que la cota para el error de Parareal es

$$\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| \leq \frac{\gamma \alpha^k}{(k+1)!} \beta^{n-k-1} \prod_{l=0}^k (n-l),$$

o bien

$$\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| \leq \frac{C_3}{C_1} \frac{(C_1 \Delta T^{p+1})^{k+1}}{(k+1)!} (1 + C_2 \Delta T)^{n-k-1} \prod_{l=0}^k (n-l).$$

Hemos obtenido el primer resultado del teorema (5.8). Para la segunda estimación, basta con tomar la cota que resulta del desarrollo de Taylor de la función exponencial:

$$(1 + C_2 \Delta T)^{n-k-1} \leq e^{C_2 \Delta T (n-k-1)} = e^{C_2 (T_n - T_{k+1})},$$

de donde

$$(C_1 \Delta T^{p+1})^{k+1} \prod_{l=0}^k (n-l) \leq \Delta T^{p(k+1)} (C_1 \Delta T)^{k+1} n^{k+1} = \Delta T^{p(k+1)} (C_1 T_n)^{k+1},$$

y obtenemos:

$$\|\mathbf{u}(T_n) - \mathbf{U}_n^k\| \leq \frac{C_3}{C_1} \frac{(C_1 T_n)^{k+1}}{(k+1)!} e^{C_2 (T_n - T_{k+1})} \Delta T^{p(k+1)}.$$

□



Este teorema es el resultado principal sobre estimación del error en el algoritmo Parareal. La primera estimación, (5.8), nos muestra como, cuando  $k = n$ , el método converge, pues el productorio contiene un término nulo. En la segunda estimación, (5.9), podemos observar que el error tiende a cero al aumentar  $k$  y que, además, el orden del método aumenta  $p$  veces en cada iteración, siendo  $p$  el orden del método grosero empleado. Este hecho fue ya advertido por Lions, Maday y Turinici en sus trabajos seminales [3].

## 5.2 Aproximación numérica de la ecuación del calor

En esta sección nos centramos en la resolución de un problema de ecuaciones en derivadas parciales con condiciones de contorno de Dirichlet. Profundizaremos en su resolución y mostraremos algunos resultados de interés en cuanto a velocidad y precisión del método. Esto último será estudiado con mayor detalle en el próximo capítulo, basado en la aplicación de Parareal.

En concreto, vamos a aplicar el método a la ecuación del calor partiendo de una ecuación cuya solución conozcamos, para así poner a prueba el algoritmo.

Consideremos  $\Omega = [0, 1] \times [0, 1]$ . Vamos a resolver el problema:

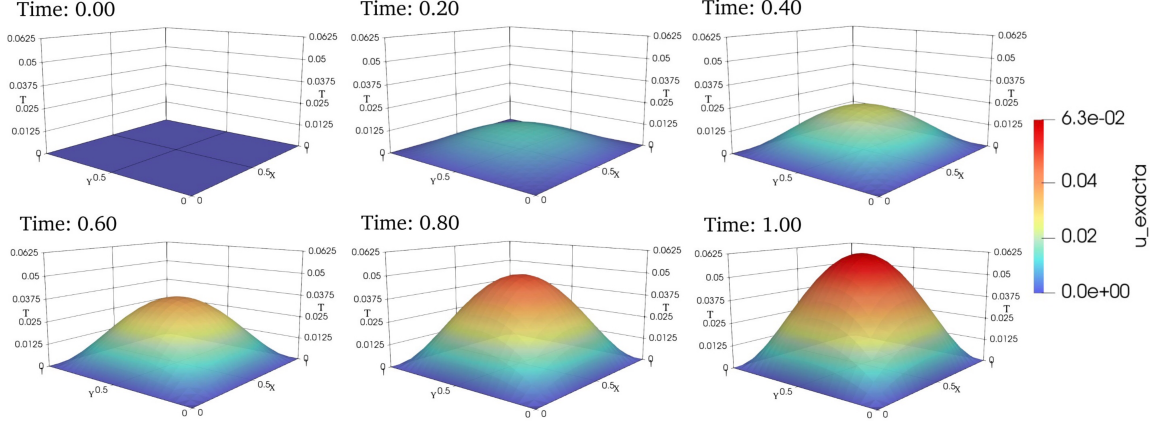
$$\begin{aligned} u_t - \Delta u &= (x^2 - x)(y^2 - y) - 2t(y^2 - y + x^2 - x), & \text{en } \Omega \times (0, 1), \\ u &= 0, & \text{en } \partial\Omega \times (0, 1), \\ u(\mathbf{x}, 0) &= 0, & \text{en } \Omega. \end{aligned} \quad (5.13)$$

Sabemos que la solución a dicho problema viene dada por la expresión

$$u(x, y, t) = x(x - 1)y(y - 1)t$$

por lo que vamos a comenzar representándola en el dominio antes considerado, para distintos valores de  $t$ :

## 5. PARAREAL



Pasamos ahora a calcular la solución numérica haciendo uso del Método de los Elementos Finitos en el dominio espacial y del método de Euler implícito en el temporal. Por simplicidad de exposición, empezaremos introduciendo la discretización en tiempo antes de la discretización total (espacio-temporal). Partimos de la ecuación del calor:

$$u_t - \Delta u = f, \quad \text{con } f(x, y, t) = (x^2 - x)(y^2 - y) - 2t(y^2 - y + x^2 - x)$$

Denotemos las iteraciones del método de Euler por  $k$ , y consideremos  $\Delta t$  el paso en tiempo. Aproximemos  $u_t$  de forma implícita:

$$\frac{u^k - u^{k-1}}{\Delta t} - \Delta u^k = f^k.$$

Multiplicando la expresión por  $\Delta t$ :

$$u^k - \Delta t \cdot \Delta u^k = \Delta t \cdot f^k + u^{k-1}.$$

Multiplicando por una función arbitraria  $v \in H^1(\Omega)$  e integrando en el dominio espacial:

$$\int_{\Omega} u^k v - \Delta t \int_{\Omega} \Delta u^k v = \Delta t \int_{\Omega} f^k v + \int_{\Omega} u^{k-1} v.$$

Aplicando integración por partes a la izquierda de la igualdad:

$$\int_{\Omega} u^k v + \Delta t \int_{\Omega} \nabla u^k \nabla v - \int_{\partial\Omega} \nabla u^k \mathbf{n} v = \Delta t \int_{\Omega} f^k v + \int_{\Omega} u^{k-1} v.$$

Como estamos tomando condiciones de contorno de Dirichlet homogéneas en toda la frontera de  $\Omega$ , tenemos que  $v \in V$  donde  $V = H_0^1(\Omega)$ . Luego,  $v = 0$  en  $\partial\Omega$  y se anula la integral en la frontera. Obtenemos:

$$\int_{\Omega} u^k v + \Delta t \int_{\Omega} \nabla u^k \nabla v = \Delta t \int_{\Omega} f^k v + \int_{\Omega} u^{k-1} v.$$

## 5.2 Aproximación numérica de la ecuación del calor

---

Así, haciendo el cambio de notación:

$$a(u, v) = \int_{\Omega} u^k v + \Delta t \int_{\Omega} \nabla u^k \nabla v \quad L(v) = \Delta t \int_{\Omega} f^k v + \int_{\Omega} u^{k-1},$$

nuestro problema se transforma en encontrar  $u \in V$  tal que

$$a(u, v) = L(v), \quad v \in V.$$

Resolvemos este problema haciendo uso del algoritmo Parareal. Para ello, consideraremos como método de resolución para **F** y **G** el Método de los Elementos Finitos en espacio y Euler implícito en tiempo que acabamos de definir, tomando un mayor número de iteraciones para **F** (la aproximación lenta pero precisa) que para **G** (la aproximación rápida pero poco rigurosa).

El número de subintervalos en que dividamos el dominio temporal, determinará el número de núcleos que utilizaremos (siempre que no sobrepase la cantidad de estos disponibles).

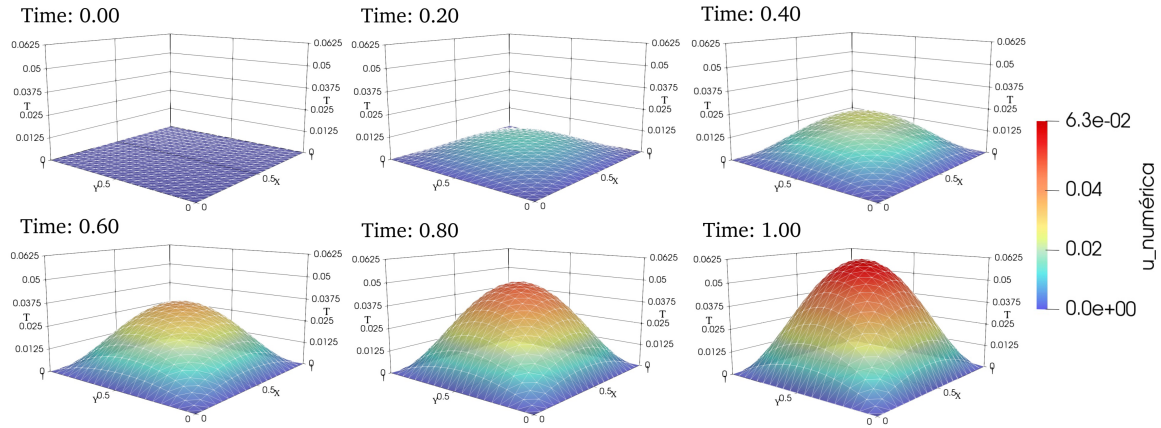
Consideraremos 10 subintervalos temporales,  $[T_n, T_{n+1}]$  con  $n \in \{0, 1, \dots, 9\}$ , siendo  $T_0 = 0$  y  $T_{10} = 1$ . Esto nos lleva a la resolución de 10 subproblemas semejantes al original, en sus respectivos dominios temporales.

Tomaremos para **F** 50 iteraciones del método, las cuales se realizarán en un núcleo para cada subproblema. Para **G**, tomaremos solamente 10, pues estas se realizan de manera secuencial, ralentizando el programa. Recordemos que tras el cálculo de las soluciones, se realiza una corrección entre **F** y **G**, por lo que nos interesa ahorrar el máximo tiempo posible en el cálculo de las soluciones mediante **G**. En resumen, los parámetros utilizados son:

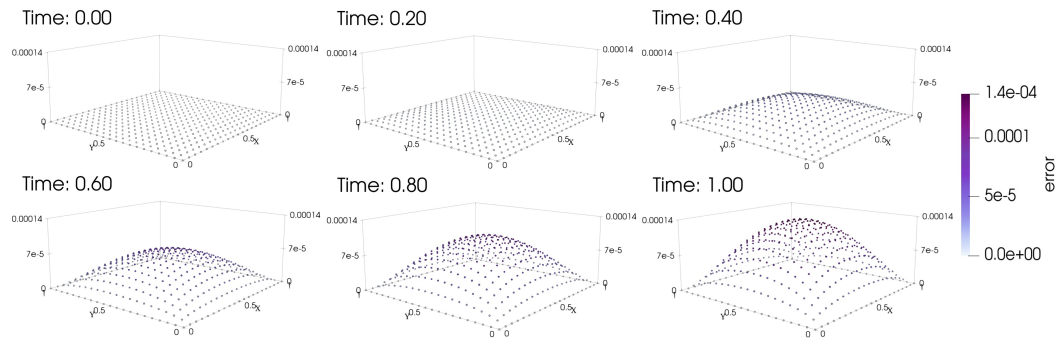
- Subintervalos = 10
- Partición del dominio espacial =  $20 \times 20$
- Iteraciones gruesas = 10
- Iteraciones finas = 50
- Iteraciones de Parareal = 5
- Procesadores = 20.

Pintamos la solución en los mismos instantes temporales que habíamos representado la exacta. En este caso, dibujaremos además la malla usada en el método:

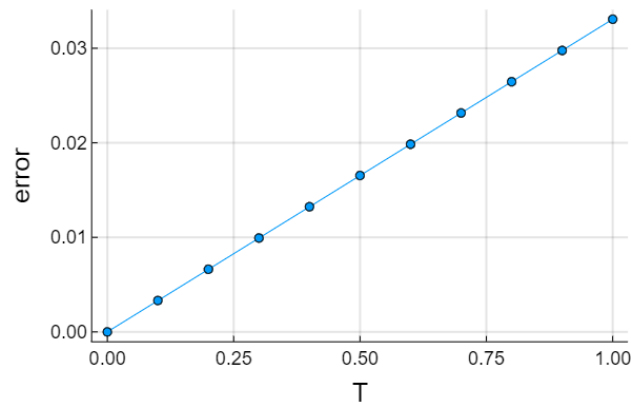
## 5. PARAREAL



Observamos que la solución obtenida se asemeja a la solución exacta del problema estudiado. Para comprobar dicha similitud vamos a representar el error cometido en cada uno de los nodos de la malla considerada en el Método de los Elementos Finitos. Veamos en primer lugar como se comporta dicho error con el paso del tiempo:



Mostramos ahora el aumento del error en norma  $L^2$ :

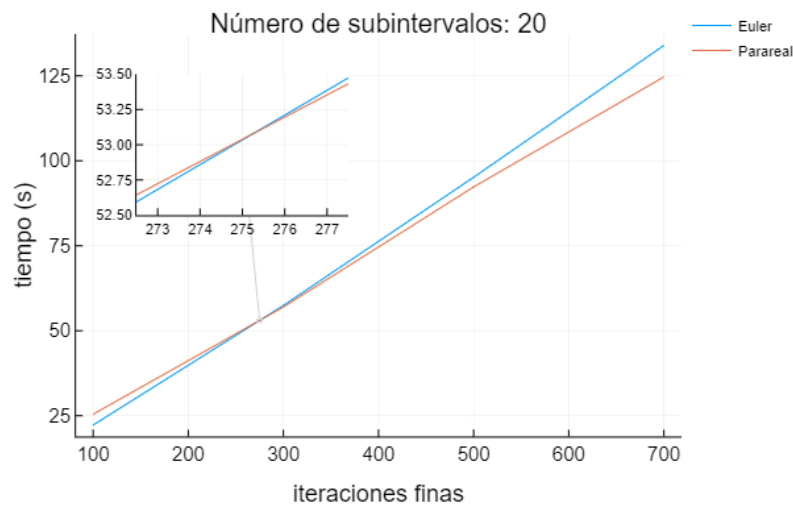


## 5.2 Aproximación numérica de la ecuación del calor

Vemos en ambas representaciones como el error va aumentando en tiempo, lo cual era de esperar debido a la acumulación de pequeños errores en cada instante. También podemos observar como la diferencia entre la solución exacta y la obtenida por el algoritmo Parareal es pequeña.

Vamos a analizar ahora los tiempos de cómputo en relación con los obtenidos por un método de Euler secuencial clásico. Por simplicidad, reduciremos los parámetros que no influyan en la diferencia entre ambos métodos, como la partición espacial para el Método de los Elementos Finitos, e iremos modificando el número de iteraciones finas realizadas. Los parámetros utilizados ahora son:

- Subintervalos = 20
- Partición del dominio espacial =  $5 \times 5$
- Iteraciones gruesas = 5
- Iteraciones finas = 100, 300, 500, 700
- Iteraciones de Parareal = 2
- Procesadores = 20.



Observamos como el algoritmo Parareal mejora el tiempo de cómputo a partir de un número determinado de iteraciones finas (en torno a 275), las cuales, a su vez, mejoran la solución conforme aumentan. Es importante recalcar que la mejora del tiempo se ve afectada por la escasez de recursos actuales para este tipo de computación. Las bibliotecas de los lenguajes que permiten resolver de manera conjunta un programa de elementos finitos y paralelo en tiempo se encuentran aún en desarrollo, por lo que se pierde rapidez en la adaptación de las soluciones a formatos compatibles con ambos. Conforme estos detalles

## **5. PARAREAL**

---

se vayan solventando, la mejora de Parareal será más y más significativa frente a los métodos secuenciales actuales. Comprobaremos esto en el siguiente capítulo, donde vamos a tratar, en primer lugar, un sistema de ecuaciones diferenciales, lo que nos va a permitir conocer Parareal sin las limitaciones de su implementación junto con los elementos finitos.

# Modelos epidemiológicos

En esta sección aplicaremos el método Parareal a modelos matemáticos basados en ecuaciones diferenciales que tratan enfermedades en una población. Comenzaremos hablando sobre qué son estos modelos, para posteriormente hacer un ejemplo en el cual estudiaremos las mejoras presentes de Parareal frente a un método secuencial. Por último, introduciremos un modelo epidemiológico con difusión espacial, un tema en el que se está investigando en la actualidad.

## 6.1 Modelos SIR

El modelo SIR es un modelo epidemiológico propuesto en 1927 por W.O. Kermack y A.G. Mckendrick [18]. En el se trata el estudio de una población bajo los efectos de una epidemia que se propaga entre los individuos. Puede encontrarse más información, por ejemplo, consultando [19]. El nombre de este modelo proviene de las categorías consideradas en el estudio:

- Susceptibles: subconjunto de la población que no presenta inmunidad frente a la epidemia considerada. En el modelo se denotará por la letra  $S$ .
- Infectados: corresponden a los individuos que presentan la enfermedad en ese instante. Los representaremos por la letra  $I$ .

## 6. MODELOS EPIDEMIOLÓGICOS

---

- Recuperados (o retirados): hace referencia a la parte de la población que ha sufrido ya la enfermedad pero actualmente no la padece y no puede ser contagiada de nuevo. Por simplicidad, dentro de este subconjunto se consideran aquellos individuos que se han recuperado y ahora son inmunes, y aquellos que han fallecido. En el modelo corresponden a la letra  $R$ .

Consideremos una población fija cuya cantidad de sujetos sea  $N$ . Nos centraremos en un modelo sencillo, por lo que no tendremos en cuenta incrementos en la población por nacimientos, ni disminuciones por muertes ajenas a la epidemia.

La variación en el número de individuos susceptibles respecto al tiempo viene condicionada por el número de sujetos sanos que se contagian, disminuyendo conforme estos últimos aumentan. Así, los susceptibles pasan a ser infectados cuando entran en contacto con contagiados. La velocidad a la que esto ocurre vendrá dada por el parámetro de transmisión de la epidemia, al cual denotaremos por  $\beta$ . Este parámetro indica la probabilidad de que un individuo susceptible se contagie de uno infectado cuando entran en contacto. De este modo, la primera ecuación diferencial del modelo viene dada por:

$$S'(t) = -\beta S(t)I(t)$$

Del mismo modo, la variación en el número de infectados dependerá de los individuos sanos que se contagian ( $\beta SI$ ), y disminuirá conforme los enfermos sanen o fallezcan. Consideraremos la tasa de recuperación junto con la de mortalidad y la denotaremos por  $\gamma$ , siendo  $\frac{1}{\gamma}$  la duración media de la enfermedad sobre un individuo. Obtenemos ahora la expresión:

$$I'(t) = \beta S(t)I(t) - \gamma I(t)$$

Por último, cabe considerar la alteración en el número de recuperados, los cuales incrementarán conforme disminuyen los infectados.

$$R'(t) = \gamma I(t)$$

Sin embargo, como estamos considerando un número de individuos fijo en la población, esta última expresión puede escribirse también como:  $R(t) = N - S(t) - I(t)$ .



Por otra parte, en el instante inicial consideraremos un número  $I(0) = I_0$  de infectados. De este modo,  $S(0) = N - I_0$ , ya que inicialmente no habrá individuos recuperados ni que hayan fallecido a causa de la enfermedad; es decir,  $R(0) = 0$ .

Un parámetro importante en el estudio de un modelo SIR es  $R_0$ , que representa el número de individuos contagiados por un infectado si toda la población es susceptible. Este valor se denomina *Tasa básica de reproducción* y tanto él como  $\gamma$  se obtienen mediante la observación, y dado que  $R_0 = \beta \frac{1}{\gamma}$ , podemos obtener también la tasa de transmisión,  $\beta$ .

El modelo SIR presenta una gran cantidad de variantes que dependen, entre otros motivos, de la enfermedad tratada. Así, podrían estudiarse enfermedades con periodos de incubación, añadir inmunidad por vacunas, considerar que un individuo pueda contagiarse en más de una ocasión, etc.

En el siguiente ejemplo nos centraremos en un modelo sencillo, definido por las ecuaciones anteriores.

**Ejemplo 6.1.** Consideremos una población de  $N = 103$  habitantes y el modelo SIR:

$$\begin{cases} S'(t) = -0.07 S(t)I(t), & t \in (0, 5] \\ I'(t) = 0.07 S(t)I(t) - 0.5 I(t), & t \in (0, 5] \\ R'(t) = 0.5 I(t), & t \in (0, 5] \end{cases} \quad (6.1)$$

con valores iniciales:

$$S(0) = 100, \quad I(0) = 3$$

Vamos a resolver este sistema haciendo uso del algoritmo Parareal y de un método secuencial equivalente. Para ello, vamos a comenzar fijando los parámetros del método. Realizaremos varias pruebas modificando dichos parámetros para obtener una mejor comparación. Además, como el tiempo necesario para resolver un sistema de ecuaciones diferenciales ordinarias es mucho menor que el que se necesita para ecuaciones en derivadas parciales y no habrá “interferencias” debidas a la aproximación espacial, este ejemplo nos servirá para poner a prueba la rapidez del algoritmo.

Comencemos fijando, para el algoritmo Parareal, un número de subintervalos igual al número de núcleos disponibles. Así, partimos de  $N = 16$  subproblemas en los subintervalos temporales  $[T_{n-1}, T_n]$ , con  $n \in \{1, \dots, 16\}$ , siendo  $T_0 = 0$  y  $T_{16} = 5$ . Al tomar esta disposición, estamos aprovechando al máximo todos los núcleos libres.

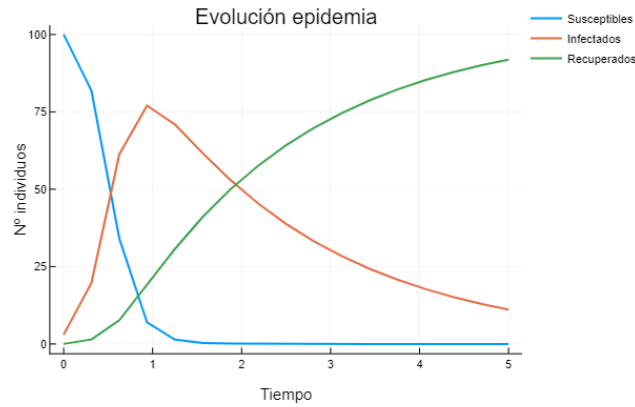
## 6. MODELOS EPIDEMIOLÓGICOS

---

Ahora, dado que queremos mostrar el potencial de la parte paralela, tomaremos un número de iteraciones para la aproximación precisa,  $F$ , mucho mayor que para  $G$ . Por ejemplo, consideraremos la primera variando entre 100,000 y 1,000,000, mientras que para la aproximación gruesa tomaremos 100. De esto modo, el algoritmo Parareal realizará de forma paralela entre 100,000 y 1,000,000 iteraciones en cada uno de los 16 intervalos, mientras que el método de resolución clásico, realizará entre  $16 \cdot 100,000$  y  $16 \cdot 1,000,000$  iteraciones de manera secuencial.

Por último, tomaremos  $k = 5$  iteraciones de Parareal; es decir, el algoritmo paralelo deberá ejecutarse 5 veces mientras el secuencial se realiza una vez. Es interesante comparar, no solo los tiempos de computo, sino también la exactitud de la solución obtenida, la cual dependerá del número de iteraciones de Parareal, alcanzándose la misma solución que la obtenida mediante Euler cuando el número de iteraciones coincida con el de subintervalos.

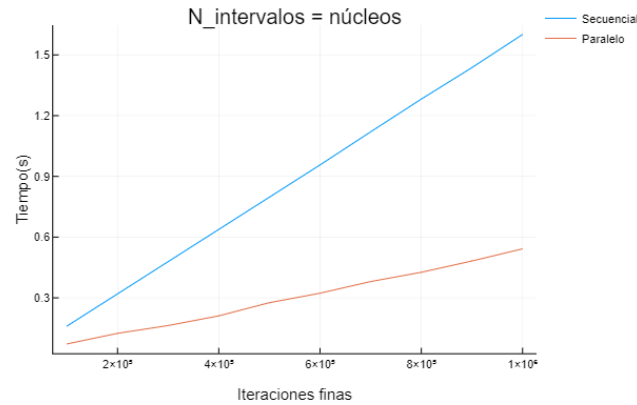
La solución obtenida para los datos expuestos anteriormente, tomando para  $F$  1,000,000 iteraciones es la siguiente:



Los errores cometidos respecto a la solución clásica mediante Euler secuencial son:

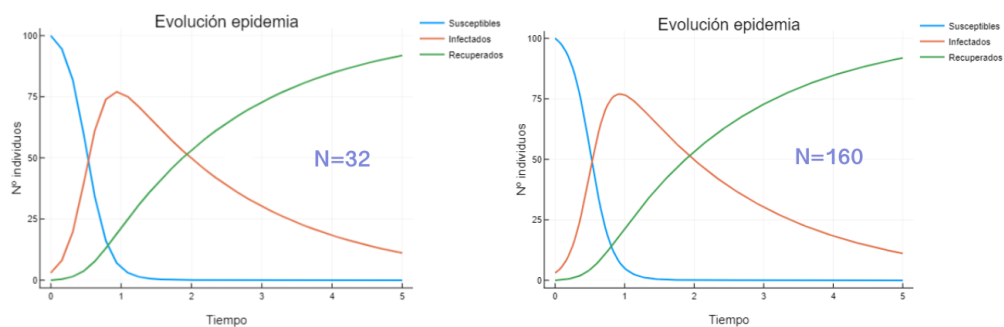
$$\text{Error absoluto} = 3,55 \cdot 10^{-13} \quad \text{Error relativo} = 3,83 \cdot 10^{-15}$$

Es decir, con 5 iteraciones del algoritmo Parareal, hemos obtenido una solución con un error ínfimo respecto a la aproximación mediante Euler. Veamos la mejora en tiempo haciendo variar el número de iteraciones finas en el rango antes mencionado.



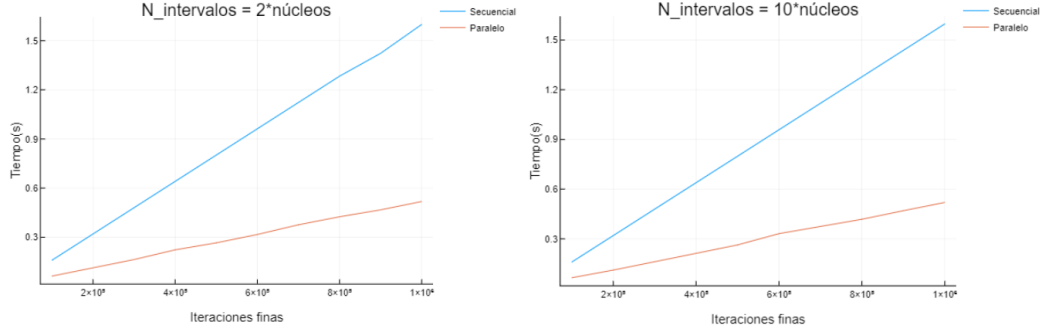
Observamos como se obtiene una mejora bastante significativa conforme aumenta el número de iteraciones finas, llegándose a triplicar el tiempo necesario en el método secuencial respecto al algoritmo Parareal.

Sin embargo, observamos que la solución que se obtiene es poco regular, resultado de la interpolación de los valores en los 16 subintervalos considerados. Por ello, nos planteamos considerar un mayor número para  $N$ , con el fin de obtener una representación de la solución con mayor exactitud en cada instante de tiempo. El problema que se nos presenta aquí es que, al considerar un mayor número de subintervalos que el de núcleos disponibles, podría ocurrir que el tiempo de espera y comunicación para realizar la siguiente tanda de iteraciones en paralelo, ralentizara el programa. Para comprobar si dicha espera es significativa, realizaremos los cálculos para  $N = 32$  (el doble de núcleos) y  $N = 160$  (10 veces el número de núcleos).



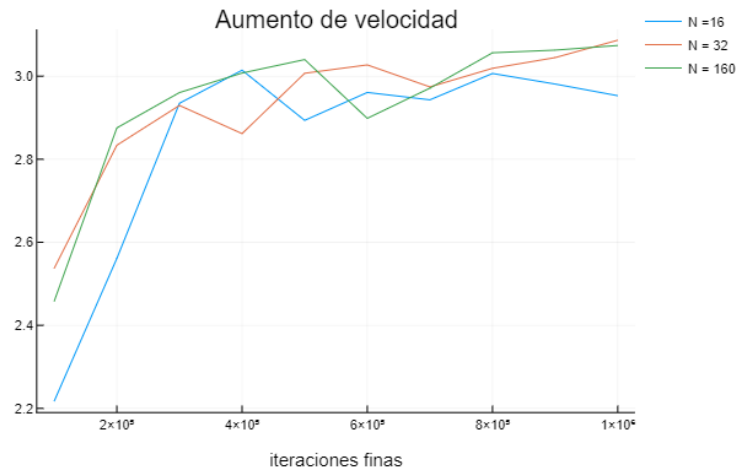
Veamos ahora si estas representaciones suponen una pérdida en la mejora de la velocidad de computo:

## 6. MODELOS EPIDEMIOLÓGICOS



Como podemos observar, el incremento de velocidad sigue siendo enorme, concluyendo así que el tiempo de espera entre tandas de cálculo paralelo es insignificante frente al tiempo necesario en los cálculos restantes.

Por último, mostremos una gráfica que representa el porcentaje de aumento en la velocidad en cada uno de los programas anteriores (16, 32 y 160 subintervalos).



La gráfica se ha obtenido dividiendo los tiempos del secuencial entre los de Parareal, por lo que obtenemos que estos eran en torno al triple de lo que tarda el algoritmo paralelo.

Para finalizar este ejemplo, vamos a comprobar que se verifica uno de los teoremas expuestos en el capítulo anterior. Veamos que, dado un número de iteraciones  $k$  y un número de subintervalos  $N$ , la solución  $U_n^k = F(T_n, 0, u^0)$  para  $k \geq n, n \in \{0, \dots, N\}$ .

Para ello, tomaremos  $k = 8$ ,  $N = 5$ , y nos centraremos en el último subintervalo. Los valores obtenidos son:

Número de subintervalos: 5			
Iteración	Susceptibles	Infectados	Recuperados
0	0.0002062767864263847	11.122882734453858	91.87691098875982
1	0.0002560906457992356	11.120721509251593	91.87902240011272
2	0.0002591186212458456	11.120538661462502	91.87920221992499
3	0.00025919154158319954	11.120538189903598	91.87920261854836
4	0.00025919227956150973	11.120538188823806	91.87920261889585
5	0.00025919228209789696	11.12053818882136	91.8792026188947
6	0.00025919228209789696	11.12053818882136	91.8792026188947
7	0.00025919228209789696	11.12053818882136	91.8792026188947
8	0.00025919228209789696	11.12053818882136	91.8792026188947

#### Solución por método preciso

Susceptibles	Infectados	Recuperados
0.00025919228209789696	11.12053818882136	91.8792026188947

Efectivamente, como podemos observar en los datos, la solución obtenida por el algoritmo Parareal coincide con la obtenida por el método fino cuando  $k \geq n$ .

Por último, nos centraremos en una variante: un modelo SIR con difusión. De esta forma, no solo analizaremos la cantidad de individuos presentes de cada categoría a lo largo del tiempo, sino también como se distribuyen en el espacio.

## 6.2 Difusión

En esta sección vamos a presentar un modelo epidemiológico que trata la evolución de una enfermedad sobre una población, no solo en tiempo, sino también en espacio. Existen pocas referencias que traten estos modelos, tratándose de un tema actualmente en investigación. Estos esquemas presentan la ventaja de poder modelar la migración espacial de los individuos, y el inconveniente de requerir un gran esfuerzo computacional, viéndose reemplazados en la práctica por los modelos usuales de ecuaciones diferenciales ordinarias. Es por ello que encontramos la paralelización de modelos SIR con difusión como un recurso sin explotar que puede presentar avances significativos en este campo. Información más detallada, así como la demostración a los resultados que se presentan a continuación, puede encontrarse, por ejemplo, en [20]. Para comenzar el estudio, introduciremos en el modelo anterior términos difusivos, que representen la propagación de las categorías en términos espaciales. Así, nuestro nuevo objeto de estudio es:

Sea  $\Omega$  un dominio acotado,  $\Omega \in \mathbb{R}^n$ , con frontera regular  $\partial\Omega$ , planteamos el sistema de EDP:

## 6. MODELOS EPIDEMIOLÓGICOS

---

$$\begin{cases} S_t - \alpha_S \Delta S = -\beta IS, & z \in \Omega, t > 0, \\ I_t - \alpha_I \Delta I = \beta IS - \nu I, & z \in \Omega, t > 0, \\ R_t - \alpha_R \Delta R = \nu I, & z \in \Omega, t > 0. \end{cases} \quad (6.2)$$

Consideraremos nula la propagación de la epidemia hacia el exterior de nuestro dominio, por lo que añadimos las condiciones de Neumann:

$$\partial_n S = \partial_n I = \partial_n R = 0, \quad z \in \partial\Omega, t > 0.$$

En cuanto a las condiciones iniciales, denotemos:

$$S(z, 0) = S_0(z) \geq 0, \quad I(z, 0) = I_0(z) \geq 0, \quad R(z, 0) = R_0(z) \geq 0, \quad z \in \overline{\Omega}.$$

Por otra parte, tomaremos una misma constante de difusión independientemente del tipo de individuo; es decir, consideraremos que toda la población tiene las mismas capacidades de movimiento difusivo. Por ello, tomaremos  $\alpha_S = \alpha_I = \alpha_R = \alpha$ .

En primer lugar, dado que estamos tratando individuos de una población, es importante que la solución al problema sea positiva. Por ello, presentamos los dos siguientes resultados.

**Lema 6.1.** Sea  $u \in \mathcal{C}(\overline{\Omega} \times [0, T]) \cap \mathcal{C}^{2,1}(\Omega \times (0, T))$  tal que:

- $u_t - \alpha \Delta u \geq v(z, t)u, \quad z \in \Omega, 0 < t < T.$
- $\frac{\partial u}{\partial n} \geq 0, \quad z \in \partial\Omega, 0 < t < T.$
- $u(z, 0) \geq 0, \quad z \in \overline{\Omega}.$

Entonces  $u(z, t) \geq 0$  para  $(z, t) \in \overline{\Omega} \times [0, T]$ . Además,  $u(z, t) \geq 0$  o  $u \equiv 0$  en  $\Omega \times [0, T]$ .

**Lema 6.2.** Cualquier solución de (6.2) con función inicial positiva, es positiva.

*Demostración.* Sea  $(S, I)$  una solución de (6.2) en  $\Omega \times (0, T)$ , donde consideramos únicamente las dos primeras incógnitas, pues la tercera puede obtenerse como  $R(z, t) = N - S(z, t) - I(z, t)$ . Entonces, por hipótesis, tenemos que

$$S(z, 0) \geq 0, I(z, 0) \geq 0, \quad \frac{\partial S}{\partial n} = \frac{\partial I}{\partial n} = 0,$$

es decir, se verifican dos de las condiciones del lema anterior. Veamos la última: consideremos  $\tau \in (0, T)$ , entonces

$$I_t - \alpha \Delta I = \beta SI - \gamma I, \quad z \in \Omega, 0 < t \leq \tau,$$

$$I_t - \alpha \Delta I = v_1 I, \quad \text{donde } v_1 = \beta S - \gamma.$$

Como  $v_1 \in \mathcal{C}(\overline{\Omega} \times [0, \tau])$ , por el lema 6.1, tenemos que  $I > 0$  en  $\Omega \times [0, \tau]$ . Análogamente,

$$S_t - \alpha \Delta S = -\beta I S, \quad \text{y sea } v = \beta I, \quad z \in \Omega, 0 < t \leq \tau.$$

Entonces,  $S > 0$  en  $\Omega \times (0, \tau)$ . Como  $\tau \in [0, t]$  es arbitrario, podemos concluir que  $S > 0, I > 0$  en  $\Omega \times [0, T]$ .  $\square$

Por último, antes de comenzar con la aplicación de Parareal al problema, presentamos un resultado que nos asegura que el tamaño de población final no superará al inicial, dado que nuestro modelo no considera nacimientos.

**Teorema 6.1.** Sea  $(S, I) \in [\mathcal{C}(\overline{\Omega} \times [0, T] \cap \mathcal{C}^{2,1}(\Omega \times (0, T)))]^2$  la solución de (6.2) con las condiciones antes consideradas. Entonces

$$0 < S(z, t) + I(z, t) \leq \max\{\|S_0(z) + I_0(z)\|_\infty, N\}.$$

Pasamos ahora a la aplicación, que centraremos en la resolución del siguiente ejemplo.

**Ejemplo 6.2.** Sea  $\Omega = [0, 1] \times [0, 1]$  y consideremos el intervalo temporal  $[0, 2]$ . El modelo a estudiar es:

$$\begin{cases} S_t - 0.02 \Delta S = -0.95 I S, & z \in \Omega, t \in [0, 2], \\ I_t - 0.02 \Delta I = 0.95 I S - 0.99 I, & z \in \Omega, t \in [0, 2], \\ R_t - 0.02 \Delta R = 0.99 I, & z \in \Omega, t \in [0, 2]. \end{cases} \quad (6.3)$$

Consideremos, además de las condiciones de Neumann antes expuestas, las condiciones iniciales:

$$\begin{aligned} S_0(x) &= 10 - 4e^{-200((x-x_0)^2 + (y-y_0)^2)} \\ I_0(x) &= 4e^{-200((x-x_0)^2 + (y-y_0)^2)} \\ R_0(x) &= 0, \end{aligned} \quad (6.4)$$

con  $x_0 = y_0 = \frac{1}{2}$ , que representan un foco de infección en el centro del dominio espacial.

Resolveremos utilizando el algoritmo Parareal, donde la solución espacial vendrá dada por el Método de los Elementos Finitos, y en el dominio temporal aplicaremos Euler semi-implícito, ya que pretendemos hacer un esquema de estructura lo más paralela posible. Desarrollemos la discretización en tiempo, para lo que mantendremos la notación del modelo (6.2) en lugar de los valores concretos por simplicidad.

Dados  $S^{n-1}, I^{n-1}$  y  $R^{n-1}$ , hallar  $S^n, I^n$  y  $R^n$  solución de:

$$\frac{S^n - S^{n-1}}{k} - \alpha \Delta S^n = -\beta I^{n-1} S^n.$$

## 6. MODELOS EPIDEMIOLÓGICOS

---

Reordenando los términos:

$$(1 + k\beta I^{n-1})S^n - k\alpha\Delta S^n = S^{n-1}.$$

Análogamente, para  $I^n$  y  $R^n$ :

$$\frac{I^n - I^{n-1}}{k} - \alpha\Delta I^n = \beta I^n S^{n-1} - \mu I^n$$

$$(1 - k\beta S^n + k\mu)I^n - k\alpha\Delta I^n = I^{n-1}$$

$$\frac{R^n - R^{n-1}}{k} - \alpha\Delta R^n = \mu I^{n-1}$$

$$R^n - k\mu I^{n-1} - k\alpha\Delta R^n = R^{n-1}.$$

La formulación variacional correspondiente es: Dados  $S^{n-1}, I^{n-1}, R^{n-1} \in H^1(\Omega)$ , hallar  $S^n, I^n, R^n \in H^1(\Omega)$  tales que

$$\int_{\Omega} (1 + k\beta I^{n-1})S^n v_S + k\alpha \int_{\Omega} \nabla S^n \nabla v_S = \int_{\Omega} S^{n-1} v_S, \quad \forall v_S \in H^1(\Omega),$$

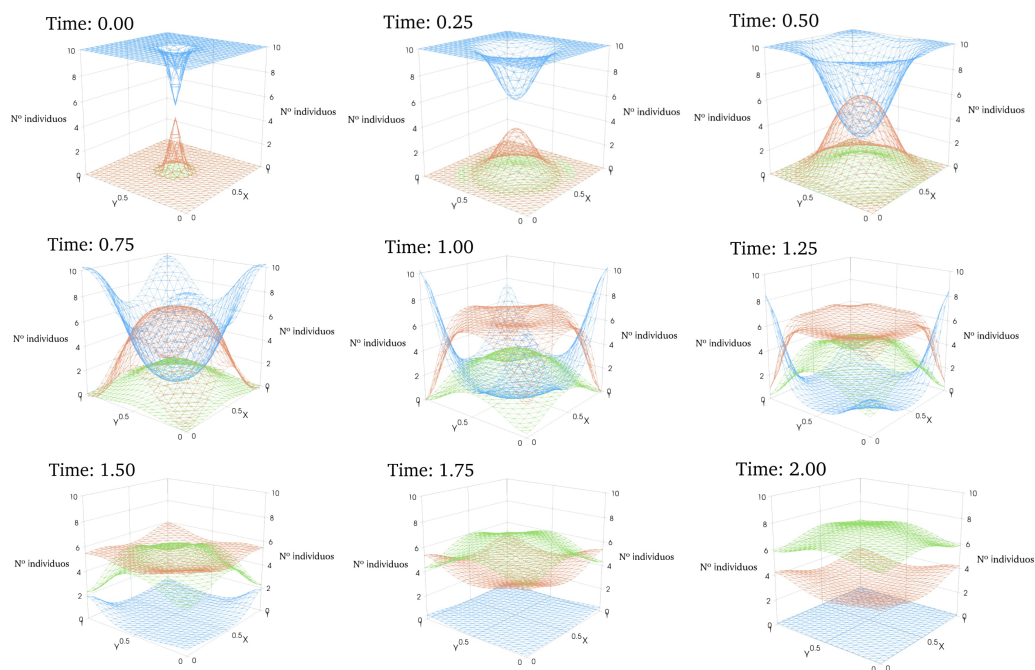
$$\int_{\Omega} (1 - k\beta S^{n-1} + k\mu)I^n v_I + k\alpha \int_{\Omega} \nabla I^n \nabla v_I = \int_{\Omega} I^{n-1} v_I, \quad \forall v_I \in H^1(\Omega),$$

$$\int_{\Omega} R^n v_R + k\mu I^{n-1} v_R + k\alpha \int_{\Omega} \nabla R^n \nabla v_R = \int_{\Omega} R^{n-1} v_R, \quad \forall v_R \in H^1(\Omega),$$

junto con las condiciones mencionadas anteriormente.

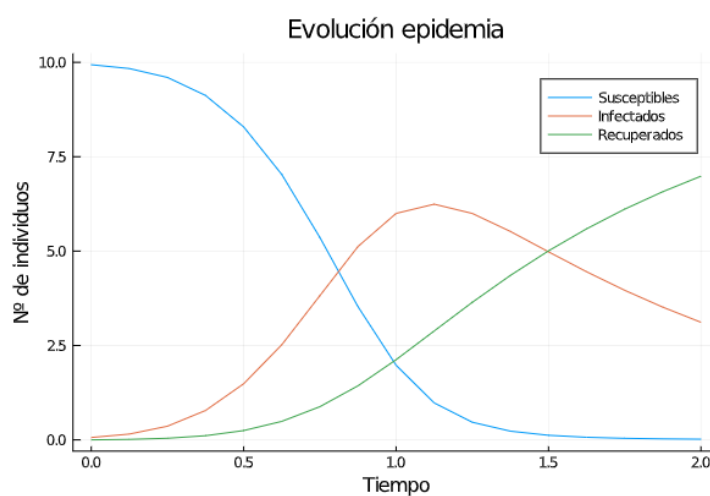
Planteamos el problema de elementos finitos  $\mathbb{P}^1$  asociado a esta formulación variacional y resolvemos en `Julia`, mediante la biblioteca `Gridap`, tomando  $N = 16$  subintervalos y  $K = 5$  iteraciones de `Parareal`. Representando las soluciones haciendo uso de `Paraview`, obtenemos:





Donde estamos representando la población susceptible de color azul, los infectados de rojo y los recuperados de verde. En estas gráficas se aprecia la difusión de los infectados, que van propagando la enfermedad a la población susceptible en zonas donde inicialmente no había individuos infectados.

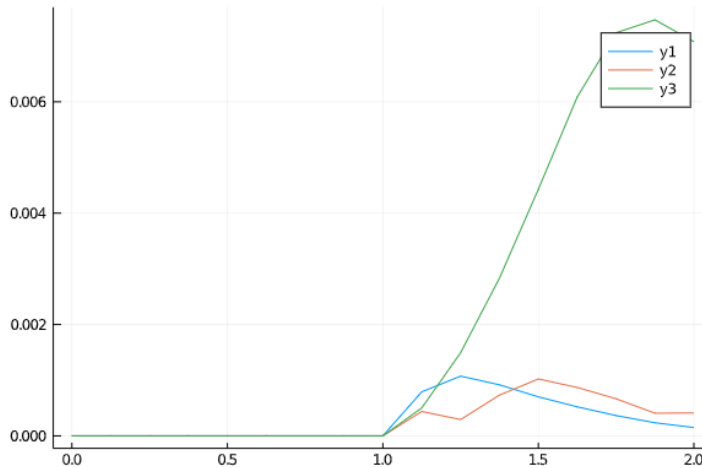
Representemos ahora la evolución de la media poblacional de cada categoría a lo largo del tiempo, para obtener una representación similar a la del modelo SIR estudiado en la sección anterior y comprobar que se mantienen los límites de población:



## 6. MODELOS EPIDEMIOLÓGICOS

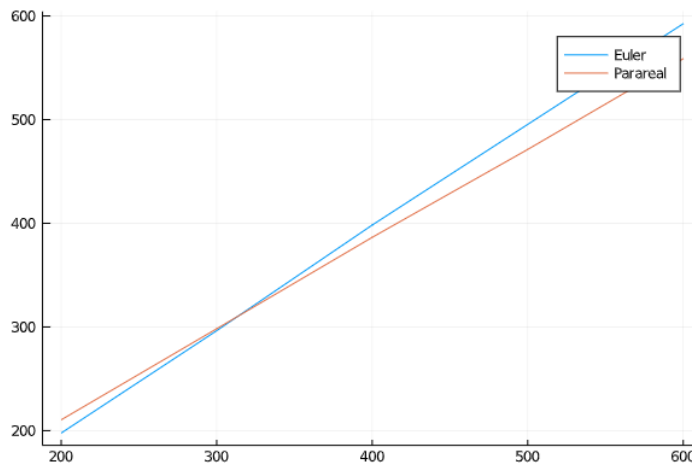
---

Veamos ahora la evolución del error en norma  $L^2$  en tiempo, comparando la solución obtenida por Parareal con la de el método de Euler secuencial.



Como podemos observar el error entre ambos métodos llega a alcanzar, como máximo, valores en torno a 0.008, que teniendo en cuenta los valores tratados, se sitúa alrededor del 0.1 %.

Por último, comprobemos si en este caso, Parareal mejora los tiempos del método secuencial. Tomaremos una partición en espacio menor, (5, 5) y dos iteraciones de Parareal para cada resolución por Euler. Fijando diez iteraciones para la aproximación grosera, obtenemos:



Como podemos observar, no encontramos mejora para las iteraciones propuestas hasta alcanzar las trescientas, caso similar al de la ecuación del calor. Un resultado que no sorprende debido a las numerosas adaptaciones que se han debido realizar para compaginar las resoluciones en tiempo y espacio, así como su introducción en arrays. El tiempo

de cómputo perdido en dichas adaptaciones, comienza a compensarse cuando es superado por la mejora que supone la división de la resolución en los diferentes núcleos. Reducir la necesidad de adaptaciones es uno de los objetivos que nos proponemos solventar en un futuro, como se mencionará en el siguiente y último capítulo.



## Conclusiones y objetivos futuros

El objetivo principal de este trabajo ha sido el algoritmo Parareal, con el fin de estudiar nuevos métodos numéricos para ecuaciones en derivadas parciales que mejorasen los tiempos de computo. Un objetivo algo ambicioso, pues en cierto sentido se adentra en los campos de investigación actuales. Se han obtenido resultados satisfactorios, llegando a alcanzar y superar las expectativas iniciales.

Hemos comenzado nuestro estudio exponiendo dos métodos en los que residía la idea de los parámetros de disparo, parte fundamental del algoritmo Parareal. Estos métodos, aún secuenciales, introducían la propuesta de una resolución partiendo de valores entonces desconocidos. El segundo de ellos, añadía una descomposición del dominio temporal en subintervalos, lo cual se hereda en el método paralelo, así como sus propiedades de convergencia.

En el capítulo 5, se ha presentado el que ha sido el algoritmo central de este trabajo: Parareal, un método de disparo, esta vez, paralelo en tiempo. Se realizaron ejemplos numéricos para la ecuación cuyos resultados fueron buenos, obteniendo una mejora en los tiempos de cómputo conforme se incrementaba el número de iteraciones precisas. A su vez,

## 7. CONCLUSIONES Y OBJETIVOS FUTUROS

---

los errores cometidos respecto a la solución exacta fueron bajos. Es por ello que en esta sección comenzamos a visualizar las ventajas de Parareal frente a un Euler secuencial clásico. Cabe destacar también los inconvenientes encontrados, situados en las limitaciones de los lenguajes de programación utilizados. El primero de ellos fue claro, en un intento de programación en `Python`. Las bibliotecas que presenta este lenguaje para paralelización en memoria compartida, como son `Numba` y `Multithreading`, manifestaban problemas a la hora de trabajar junto con `Fenics`, biblioteca inicialmente escogida para la resolución de elementos finitos. Tras varios intentos, se decidió abandonar esta idea y trasladar el estudio a un nuevo lenguaje, `Julia`. En este caso se encontró una mejor preparación para la computación paralela, aún con sus inconvenientes por la novedad del tema en estudio y del propio lenguaje. Entre otros impedimentos, se encontraron problemas relacionados con la dificultad de conjugar un algoritmo paralelo en tiempo como Parareal, con la biblioteca que permitía hacer uso de elementos finitos. Dichos problemas, presentes también en los siguientes capítulos, ralentizan los programas, pues hacen necesario el uso de cambios de formato en cada operación. Este es uno de los inconvenientes que nos proponemos solventar en un futuro, ya que dicha mejora aumentaría significativamente la rapidez del método en el caso de ecuaciones en derivadas parciales.

En el capítulo 6 pudimos probar Parareal sin la necesidad de las adaptaciones para programar elementos finitos, pues se abordó el objetivo de resolver un sistema de ecuaciones diferenciales. En concreto, se estudió el modelo SIR, obteniendo, en el mejor de los casos, una espera tres veces menor frente a los métodos secuenciales. Los óptimos resultados obtenidos en dicho ejemplo traen consigo la motivación en búsqueda de correcciones que permitan trabajar con la misma comodidad y eficiencia en el caso de las ecuaciones en derivadas parciales. Al requerir estos últimos problemas mayores tiempos de computo, las mejoras al aplicar Parareal serían más significativas, llegando a ahorrar grandes esperas si de problemas extensos se tratase.

Esto sería especialmente interesante en el último caso estudiado, el modelo epidemiológico con difusión desarrollado en esta memoria. Parte de su importancia reside en el avance en un campo que, hasta donde nosotros conocemos, se encuentra inexplorado: métodos paralelos en tiempo para modelos de epidemia con difusión. Como cabía esperar tras los resultados obtenidos en la sección anterior para la ecuación del calor, este modelo pre-

---

senta mejora a partir de un número de iteraciones determinado.

Es por estos motivos que llegamos a la conclusión de que los algoritmos paralelos en tiempo pueden ocasionar grandes mejoras para la resolución de problemas de ecuaciones en derivadas parciales. Esto conlleva que nuestro principal objetivo posterior a este trabajo sea, además de la continuación en el estudio de Parareal, la adaptación o creación de bibliotecas para el Método de los Elementos Finitos, compatibles con la programación paralela.

Así como en este trabajo se ha hecho uso de los elementos finitos en espacio y Euler en tiempo, el algoritmo propuesto admite distintos métodos, abriendo las puertas a una nueva forma de resolución notablemente más rápida.

Como Nievergelt en 1964, encontramos en el campo de la paralelización en tiempo una gran oportunidad para el cálculo numérico.







# Código Julia

En este apéndice, incluiremos los códigos de `Julia` que hemos empleado para la resolución numérica en los capítulos 5 y 6.

## A.1 Ecuación del calor

En primer lugar, expondremos el código utilizado en la implementación de la solución exacta, conocida de antemano, y la resolución numérica de la ecuación del calor, ejemplo del capítulo 5.

### A.1.1 Solución exacta

```
#Comprobamos el número de núcleos disponibles antes de comenzar
Threads.nthreads()

using Gridap
import Gridap: ▽

#En primer lugar vamos a definir la expresión de la solución ya conocida

u_exacta(x) = (x[1]*x[1]-x[1])*(x[2]*x[2]-x[2])*t

#Mallado
const global domain = (0,1,0,1)
const global partition = (20,20)
```

## A. CÓDIGO JULIA

---

```
model = CartesianDiscreteModel(domain,partition)
model=simplexify(model)
const global trian = Triangulation(model)
const global degree = 1
const global quad = CellQuadrature(trian,degree)
const global N_intervalos = 10;

const global dt_ex = 1/N_intervalos;

for i=1:N_intervalos+1
    t=(i-1)*dt_ex
    writevtk(trian, "Sol_exacta$(lpad(i,2,'0'))",
        cellfields=["u_exacta" => u_exacta])
end
```

### A.1.2 Solución numérica

```
#Parámetros
t = 0.
g(x) = 0 #valor en la frontera
f(x) = (x[1]*x[1]-x[1])*(x[2]*x[2]-x[2]) - 2*(x[2]*x[2]-x[2])*t
        - (x[1]*x[1]-x[1])*2*t #fuente de calor
u0(x) = 0. #valor inicial

const global t_init = 0.
const global t_end = 1.
const global n_iter = 5;
const global T = LinRange(t_init, t_end, N_intervalos+1)

#Espacio de funciones
const global order = 1
V0 = TestFESpace(
    reffe=:Lagrangian, order=order, valuetype=Float64,
    conformity=:H1, model=model, dirichlet_tags="boundary")

U = TrialFESpace(V0,g)

#Resolución por Euler implícito en tiempo y elementos finitos en espacio
function EulerMEF(t_init,t_end,n_t,u0)
    dt = (t_end - t_init)/n_t
    u_sol = u0
    t = t_init
    f(x) = (x[1]*x[1]-x[1])*(x[2]*x[2]-x[2]) - 2*(x[2]*x[2]-x[2])*t
            - (x[1]*x[1]-x[1])*2*t

    a(u,v) = dt * beta* ∇ (v)* ∇ (u) + u*v
    b(v) = v*dt*f + v*u0

    for i=1:n_t
        t += dt
```

```

    t_Ω = AffineFETerm(a,b, trian, quad)
    op = AffineFEOperator(U,V0,t_Ω)
    u_sol = solve(op)

    u0 = u_sol
end
return u_sol
end

#Número de iteraciones de las aproximaciones fina y gruesa

t_n_f = 50;
const global t_n_c = 10;

@inline F(t1, t0, u0) = EulerMEF(t0, t1, t_n_f,u0)
@inline G(t1, t0, u0) = EulerMEF(t0, t1, t_n_c,u0)

#Función de Gridap con el valor inicial
f2(x) = 0

const global U_u0 = TrialFESpace(V0,g)

a_u0(u,v) = u*v
b_u0(v) = v*f2

const global t_Ω_u0 = AffineFETerm(a_u0,b_u0, trian, quad)
const global op_u0 = AffineFEOperator(U_u0,V0,t_Ω_u0)

const global u0_guardar = solve(op_u0)

#Función de Parareal

function EDP_parareal()
    U =Array{Gridap.Geometry.GenericCellField{true},2}(undef, N_intervalos+1,
        n_iter+1)
    F_sol =Array{Gridap.Geometry.GenericCellField{true},1}(undef, N_intervalos+1)

    # 1.a) Inicialización (aproximción grosera)
    U[1,1] = 1*u0_guardar

    for n=1:N_intervalos
        U[n+1,1] = 1*G( T[n+1],T[n],U[n,1] )
    end

    # 1.b) Inicialización etapas parareal
    @inbounds Threads.@threads for k=1:n_iter
        U[1,k+1] = 1*u0_guardar
    end

    # 2) Bucle parareal
    for k=1:n_iter

        # 2.a) Aproximación fina (paralela) en cada subintervalo
        @inbounds Threads.@threads for n = 1:N_intervalos
            F_sol[n] = 1*F( T[n+1], T[n], U[n,k] )

```

## A. CÓDIGO JULIA

---

```
end

# 2.b) Corrección secuencial
@inbounds for n = 1:N_intervalos
    U[n+1, k+1] = F_sol[n] + G( T[n+1], T[n], U[n,k+1] )
    - G( T[n+1], T[n], U[n,k] )
end
end

return U

end

#Función completa de Euler para comparaciones
function EulerMEF2(t_init,t_end,n_t,u0,t_n_f)
    U_save = Array{Gridap.Geometry.GenericCellField{true},1}(undef,
        N_intervalos+1)
    U_save[1] = 1*u0_guardar

    cont = 2

    dt = (t_end - t_init)/n_t
    u_sol = u0
    t = t_init
    f(x) = (x[1]*x[1]-x[1])*(x[2]*x[2]-x[2]) - 2*(x[2]*x[2]-x[2])*t
        - (x[1]*x[1]-x[1])*2*t

    a(u,v) = dt * beta* ∇ (v)* ∇ (u) + u*v
    b(v) = v*dt*f + v*u0

    for i=1:n_t
        t += dt
        t_Ω = AffineFETerm(a,b, trian, quad)
        op = AffineFEOperator(U,V0,t_Ω)
        u_sol = solve(op)
        if i % t_n_f == 0
            U_save[cont] = 1*u_sol
            cont += 1
        end
        u0 = u_sol
    end
    return U_save
end

iteraciones_f = t_n_f*N_intervalos

#Guarda la solución en ficheros para Paraview
for i=1:N_intervalos+1
    sol = U2[i,end]
    writevtk(trian, "Sol_numerica$(lpad(i,2,'0'))", cellfields=["sol" => sol])
end

for i=1:N_intervalos+1
    t=(i-1)*dt_ex
    e = u_exacta - U2[i,end]
    writevtk(trian, "Error_t$(lpad(i,2,'0'))", cellfields=["e" => e])
end
```

```

end

#Error en norma L2 para cada subintervalo temporal

l2(w) = w*w

error_L2 = Array{Any,1}(undef, N_intervalos+1);

for i=1:N_intervalos+1
    el2 = sqrt(sum( integrate(l2(U2[i,end]-u_exacta), trian, quad) ))
    error_L2[i] = el2
end

#Representamos el error en norma L2 en evolución temporal
using Plots

plotly()

plot(T,error_L2,
     shape=:auto,
     xlabel="T",ylabel="error", tickfont=(13, :black),legend = nothing,
     guidefont = (18, :black),gridlinewidth=2)

#Guardamos los errores cometidos en cada iteración
for i=1:n_iter+1
    t=1.
    e = u_exacta - U2[end,i]
    writevtk(trian,"Error_k$(lpad(i,2,'0'))",cellfields=["e" => e])
end

#Comenzamos comparando los tiempos con 8 subintervalos, 16 núcleos

time_8= Array{Float64,2}(undef, 5,2);
for i=1:5
    t_n_f = 50 +10*(i-1)
    tiempo = @elapsed EDP_secuencial()
    time_8[i,1] = tiempo
    tiempo = @elapsed EDP_parareal()
    time_8[i,2] = tiempo
end

time_array=LinRange(50, 100, 5)
plotly()
plot(time_array, time_8, label = ["secuencial" "paralelo"],
     title = "N_intervalos = mitad de núcleos", xlabel="iteraciones finas",
     ylabel="tiempo (s)")

#Realizamos la misma comparación, ahora con 16 núcleos
time_16= Array{Float64,2}(undef, 5,2);
for i=1:5
    t_n_f = 50 +10*(i-1)
    tiempo = @elapsed EDP_secuencial()
    time_16[i,1] = tiempo

```

## A. CÓDIGO JULIA

---

```
    tiempo = @elapsed EDP_parareal()
    time_16[i,2] = tiempo
end

plotly()
plot(time_array, time_16, label = ["secuencial" "paralelo"],
     title = "N_intervalos = núcleos", xlabel="iteraciones finas",
     ylabel="tiempo (s)")

#Por último, comparamos con 20 subintervalos
time_20= Array{Float64,2}(undef, 4,2);
for i=1:4
    t_n_f = 100 +200*(i-1)
    iteraciones_f = t_n_f*N_intervalos
    tiempo = @elapsed EulerMEF2(t_init, t_end,iteraciones_f,u0_guardar,t_n_f)
    time_20[i,1] = tiempo
    tiempo = @elapsed EDP_parareal()
    time_20[i,2] = tiempo
end

time_array=LinRange(100, 700, 4)

begin
    plot(time_array, time_20, label = ["Euler" "Parareal"],
         title = "Número de subintervalos: 20", xlabel="iteraciones finas",
         ylabel="tiempo (s)", tickfont=(11,:black),guidelfont = (13, :black),
         legendfont=(15,:black))
    lens!([272.5, 277.5], [52.5, 53.5],
          inset = (1, bbox(0.1, 0.1, 0.35, 0.35, :top, :left)))
end
```

## A.2 Modelo SIR

En esta sección mostraremos el código empleado en la primera parte del capítulo 6, en el modelo SIR.

```
# Parámetros globales
const global N_ind = 103 # Número de individuos
const global beta = 0.07
const global nu = 0.5

const global t_init = 0.
const global t_end = 5.
t_n_f = 1000000
const global t_n_c = 100

const global N_intervalos = 10;
const global T = LinRange(t_init, t_end, N_intervalos+1)

const global n_iter = 12 # Número de iteraciones de Parareal
```

```

#Valores iniciales
const global S0 = 100.
const global I0 = 3.
const global R0 = N_ind-S0-I0
const global U0 = [S0, I0, R0]

#Ecuaciones del sistema
SIR_1(S,I) = -beta*S*I
SIR_2(S,I) = beta*S*I-nu*I
SIR_3(S,I) = nu*I

# Resuelve el sistema de EDO U' = F(U,t), U(0)=U0 en el rango
# de tiempo T = [t_0, ..., t_n] mediante el método de Euler explícito
# Almacena la solución en U_sol
function Euler_SIR(U0, t_init, t_end, n_t)
    dt = (t_end-t_init)/n_t
    t = t_init

    #println("n_t=$n_t, t_init=$t_init, t_end=$t_end, dt=$dt")
    # U_sol = zeros(size(U0))
    S0, I0, R0 = U0
    S1, I1, R1 = U0
    for i=1:n_t
        # Denotamos: U=solución en la etapa actual, U0=sol. etapa anterior
        S1 = S0 + dt*SIR_1(S0, I0)
        I1 = I0 + dt*SIR_2(S0, I0)
        R1 = R0 + dt*SIR_3(S0, I0)

        # Preparamos siguiente iteración
        t += dt
        S0, I0, R0 = S1, I1, R1

        #println("Iter $i, t=$t, u=$U_sol, ex_sol=$(exp(t))")
    end
    return [S1, I1, R1]
end

@inline F(t1, t0, u0) = Euler_SIR(u0, t0, t1, t_n_f)
@inline G(t1, t0, u0) = Euler_SIR(u0, t0, t1, t_n_c)
#F!(t1, t0, u0, U_sol) = Euler_SIR!(u0, t0, t1, t_n_f, U_sol)
#G!(t1, t0, u0, U_sol) = Euler_SIR!(u0, t0, t1, t_n_c, U_sol)

#Función Parareal
function SIR_parareal()
    U = Array{Float64,3}(undef, N_intervalos+1, n_iter+1, 3);
    Fn = Array{Float64,2}(undef, N_intervalos+1, 3);
    Un = Array{Float64,2}(undef, N_intervalos+1, 3);

    # 1.a) Inicialización (aproximación grosera)
    U[1,1,:] = U0

    for n=1:N_intervalos
        U[n+1,1,:] = G( T[n+1],T[n],U[n,1,:] )
    end
end

```

## A. CÓDIGO JULIA

---

```
# 1.b) Inicialización etapas parareal
@inbounds Threads.@threads for k=1:n_iter
    U[1,k+1,:] = U0
end

# 2) Bucle parareal
for k=1:n_iter

    # 2.a) Aproximación fina (paralela) en cada subintervalo
    begin
        @inbounds Threads.@threads for n = 1:N_intervalos
            t0 = T[n]
            t1 = T[n+1]
            #Unk = U[n,k,:]
            Fn[n,:] = F( t1, t0, U[n,k,:])
        end
    end

    # 2.b) Corrección secuencial
    for n = 1:N_intervalos
        U[n+1, k+1, :] = Fn[n,:] + G( T[n+1], T[n], U[n,k+1,:] )
        - G( T[n+1], T[n], U[n,k,:] )
    end

end

return U

end

U = SIR_parareal();

#Representamos la evolución de la epidemia
y = U[:,end,:];
x = T
plot(x, y, label = ["Susceptibles" "Infectados" "Recuperados"], xlabel="Tiempo",
    ylabel="N° individuos",title="Evolución epidemia",linewidth = 2)

#Calculamos los errores respecto a Euler tradicional
N_total = N_intervalos*t_n_f
@time resultado = Euler_SIR(U0, t_init, t_end, N_total);

dif = resultado - U[end,end,:]
error_absoluto = sqrt( sum(dif.^2) )
error_relativo = error_absoluto / sqrt(sum(resultado.^2))
println("Diferencia entre el método de Euler y
    Parareal/Euler en la última iteración:")
println("Error absoluto: $error_absoluto.\nError relativo: $error_relativo")

#Comparación tiempos
lista_tiempos_x = LinRange(100000,1000000,10)
lista_tiempos = Array{Float64,2}(undef, 10, 2);
for i=1:10
    t_n_f = 100000*i
    N_total = N_intervalos*t_n_f
```



```

t1_l = @elapsed Euler_SIR(U0, t_init, t_end, N_total);
t2_l = @elapsed SIR_parareal()
lista_tiempos[i,1] = t1_l
lista_tiempos[i,2] = t2_l
end

plotly()
plot(lista_tiempos_x, lista_tiempos, label=["Secuencial" "Paralelo"],
      title="N_intervalos = núcleos", xlabel="Iteraciones finas",
      ylabel="Tiempo (s) ")

diferencia_16 = lista_tiempos[:,1] - lista_tiempos[:,2] ;

diferencia_relativa_16 = lista_tiempos[:,1] ./ lista_tiempos[:,2] ;

plotly()
plot(lista_tiempos_x, diferencia_relativa_16, label="(Secuencial - Paralelo)",
      title="N_intervalos = núcleos", xlabel="Iteraciones finas",
      ylabel="Tiempo (s) ")

lista_tiempos_32 = Array{Float64,2}(undef, 10, 2);
for i=1:10
    t_n_f = 100000*i
    N_total = N_intervalos*t_n_f
    t1_l = @elapsed Euler_SIR(U0, t_init, t_end, N_total);
    t2_l = @elapsed SIR_parareal()
    lista_tiempos_32[i,1] = t1_l
    lista_tiempos_32[i,2] = t2_l
end

plotly()
plot(lista_tiempos_x, lista_tiempos_32, label=["Secuencial" "Paralelo"],
      title="N_intervalos = 2*núcleos", xlabel="Iteraciones finas",
      ylabel="Tiempo (s) ")

diferencia_32 = lista_tiempos_32[:,1] - lista_tiempos_32[:,2] ;

diferencia_relativa_32 = lista_tiempos_32[:,1] ./ lista_tiempos_32[:,2] ;

lista_tiempos_160 = Array{Float64,2}(undef, 10, 2);
for i=1:10
    t_n_f = 100000*i
    N_total = N_intervalos*t_n_f
    t1_l = @elapsed Euler_SIR(U0, t_init, t_end, N_total);
    t2_l = @elapsed SIR_parareal()
    lista_tiempos_160[i,1] = t1_l
    lista_tiempos_160[i,2] = t2_l
end

plotly()
plot(lista_tiempos_x, lista_tiempos_160, label=["Secuencial" "Paralelo"],
      title="N_intervalos = 10*núcleos", xlabel="Iteraciones finas",
      ylabel="Tiempo (s) ")

```

## A. CÓDIGO JULIA

---

```
diferencia_160 = lista_tiempos_160[:,1] - lista_tiempos_160[:,2] ;

diferencia_relativa_160 = lista_tiempos_160[:,1] ./ lista_tiempos_160[:,2] ;

#Comparación relativa de los tiempos
comparacion = Array{Float64,2}(undef, 10,3)
comparacion[:,1], comparacion[:,2], comparacion[:,3] =
    diferencia_relativa_16, diferencia_relativa_32, diferencia_relativa_160

plotly()
percent = 100 .* comparacion
plot(lista_tiempos_x, comparacion, label=["N =16" "N = 32" "N = 160"],
    title="Aumento de velocidad", xlabel="iteraciones finas")
```

### A.3 Modelo SIR con difusión

Por último, en esta sección presentamos el código empleado en la resolución del último ejemplo, el modelo SIR con difusión del capítulo 6.

```
Threads.nthreads()

using Gridap
import Gridap: ▽
using LinearAlgebra

#Mallado
const global domain = (0,1,0,1)
const global partition = (20,20)
model = CartesianDiscreteModel(domain,partition)
model = simplexify(model)
const global trian = Triangulation(model)
const global degree = 1
const global quad = CellQuadrature(trian,degree)

#Parámetros
const global beta = 0.95
const global nu = 0.99
const global alpha = 0.02

const global t_init = 0.
const global t_end = 2.

const global N_intervalos = 16;

const global n_iter = 5;
const global T = LinRange(t_init, t_end, N_intervalos+1);

#Espacio de funciones
```

```

const global V = TestFESpace(
    reffe=:Lagrangian, conformity=:H1, valuetype=Float64,
    model=model, order=1)

const global W = TrialFESpace(V)

function project(q, trian, quad, order)

    a(u,v) = u*v
    l(v) = v*q
    t_Ω = AffineFETerm(a,l, trian, quad)

    V = TestFESpace(
        reffe=:Lagrangian, valuetype=Float64, order=order,
        triangulation=trian, conformity=:L2)

    U = TrialFESpace(V)
    op = AffineFEOperator(U,V,t_Ω)
    qh = solve(op)
    qh
end

#La formula de cuadratura para la forma bilineal int(u*v), cambia el orden por
#ser producto de dos polinomios (cada uno de orden "degree")
const global quadL2 = CellQuadrature(trian, 2*degree)

#Funciones iniciales

x0, y0 = 0.5, 0.5

S0(x) = 10 - 4*exp(-200*((x[1]-x0)^2 + (x[2]-y0)^2))
I0(x) = 4*exp(-200*((x[1]-x0)^2 + (x[2]-y0)^2))
R0(x) = 0

S0_sol = project(S0, trian, quadL2, 1)
I0_sol = project(I0, trian, quadL2, 1)
R0_sol = project(R0, trian, quadL2, 1)

const global U0 = [S0_sol, I0_sol, R0_sol]

function EulerMEF(t_init, t_end, n_t, u0)
    S0, I0, R0 = u0
    dt = (t_end - t_init)/n_t
    t = t_init

    a_s(S,v) = (1 + dt*beta*I0)*S*v + dt*alpha* ∇ (v) * ∇ (S)
    b_s(v) = v*S0

    a_i(I,v) = (1 - dt*beta*S0 + dt*nu)*I*v + dt * alpha* ∇ (v) * ∇ (I)
    b_i(v) = v*I0

    a_r(R,v) = R*v + dt * alpha* ∇ (v) * ∇ (R)
    b_r(v) = v*(R0 + dt* nu*I0)

```

## A. CÓDIGO JULIA

---

```
for i=1:n_t
    t += dt

    t_Ω = AffineFETerm(a_s,b_s, trian, quad)
    op = AffineFEOperator(W,V,t_Ω)
    S_sol = solve(op)

    t_Ω = AffineFETerm(a_i,b_i, trian, quad)
    op = AffineFEOperator(W,V,t_Ω)
    I_sol = solve(op)

    t_Ω = AffineFETerm(a_r,b_r, trian, quad)
    op = AffineFEOperator(W,V,t_Ω)
    R_sol = solve(op)

    S0 = S_sol
    I0 = I_sol
    R0 = R_sol

end
return [S0, I0, R0]
end

#Número de iteraciones de las aproximaciones fina y gruesa

t_n_f = 70;
const global t_n_c = 20;

@inline F(t1, t0, u0) = EulerMEF(t0, t1, t_n_f,u0)
@inline G(t1, t0, u0) = EulerMEF(t0, t1, t_n_c,u0)

#Función Parareal

function EDP_parareal()
    U =Array{Gridap.Geometry.GenericCellField{true},3}(undef, N_intervalos+1,
        n_iter+1, 3)
    Fn =Array{Gridap.Geometry.GenericCellField{true},2}(undef, N_intervalos+1, 3)
    Gn =Array{Gridap.Geometry.GenericCellField{true},2}(undef, N_intervalos+1, 3)
    G0 =Array{Gridap.Geometry.GenericCellField{true},2}(undef, N_intervalos+1, 3)

    # 1.a) Inicialización (aproximción grosera)
    U[1,1,1] = 1*S0_sol
    U[1,1,2] = 1*I0_sol
    U[1,1,3] = 1*R0_sol

    for n=1:N_intervalos
        G0[n+1,1], G0[n+1, 2], G0[n+1, 3] = 1*( F(t[n+1],T[n],U[n,1,:]) )
        U[n+1,1,:] = G0[n+1,:]
    end

    # 1.b) Inicialización etapas parareal
```

```

@inbounds Threads.@threads for k=1:n_iter
    U[1,k+1,1], U[1,k+1,2], U[1,k+1,3] = 1*S0_sol, 1*I0_sol, 1*R0_sol
end

# 2) Bucle parareal
for k=1:n_iter

    # 2.a) Aproximación fina (paralela) en cada subintervalo
    @inbounds Threads.@threads for n = 1:N_intervalos
        Fn[n,1], Fn[n,2], Fn[n,3] = 1*F( T[n+1], T[n], U[n,k,:] )
    end

    # 2.b) Corrección secuencial
    @inbounds for n = 1:N_intervalos
        Gn[n+1,1], Gn[n+1, 2], Gn[n+1, 3] = 1*G( T[n+1],T[n],U[n,k+1,:] )
        U[n+1, k+1, 1] = Fn[n,1] + Gn[n+1,1] - G0[n+1,1]
        U[n+1, k+1, 2] = Fn[n,2] + Gn[n+1,2] - G0[n+1,2]
        U[n+1, k+1, 3] = Fn[n,3] + Gn[n+1,3] - G0[n+1,3]
    end
    G0 = Gn
end

return U

end

#Cálculo de la solución

U_SIR = EDP_parareal();

#Gráficas desde Jupyter y ficheros para Paraview

using Plots

suma_S = [ sum(integrate(U_SIR[i,end,1],trian,quad)) for i in 1:N_intervalos+1]
suma_I = [ sum(integrate(U_SIR[i,end,2],trian,quad)) for i in 1:N_intervalos+1]
suma_R = [ sum(integrate(U_SIR[i,end,3],trian,quad)) for i in 1:N_intervalos+1];

vector_sumas = [suma_S,suma_I, suma_R ];
vector_x = 1:N_intervalos;

plot(T, vector_sumas, label = ["Susceptibles" "Infectados" "Recuperados"],
title="Evolución epidemia", xlabel="Tiempo", ylabel="N de individuos")

for i=1:N_intervalos+1
    sol = U_SIR[i,end,1]
    writevtk(trian,"Susceptibles$(lpad(i,2,'0'))",cellfields=["sol" => sol])
end

for i=1:N_intervalos+1
    sol = U_SIR[i,end,2]
    writevtk(trian,"Infectados$(lpad(i,2,'0'))",cellfields=["sol" => sol])
end

for i=1:N_intervalos+1
    sol = U_SIR[i,end,3]

```

## A. CÓDIGO JULIA

---

```
writevtk(trian, "Recuperados$(lpad(i,2,'0'))", cellfields=["sol" => sol])
end

#Método de Euler para la comparación en tiempo y errores

function EulerMEF2(t_init,t_end,n_t,t_n_f,u0)
    U = Array{Gridap.Geometry.GenericCellField{true},2}(undef, N_intervalos+1, 3)
    S0, I0, R0 = u0
    dt = (t_end - t_init)/n_t
    t = t_init

    U[1,1] = 1*S0_sol
    U[1,2] = 1*I0_sol
    U[1,3] = 1*R0_sol

    a_s(S,v) = (1 + dt*beta*I0)*S*v + dt*alpha* ∇ (v)* ∇ (S)
    b_s(v) = v*S0

    a_i(I,v) = (1 - dt*beta*S0 + dt*nu)*I*v + dt * alpha* ∇ (v)* ∇ (I)
    b_i(v) = v*I0

    a_r(R,v) = R*v + dt * alpha* ∇ (v)* ∇ (R)
    b_r(v) = v*(R0 + dt* nu*I0)

    cont = 2

    for i=1:n_t
        t += dt

        t_Ω = AffineFETerm(a_s,b_s,trian,quad)
        op = AffineFEOperator(W,V,t_Ω)
        S_sol = solve(op)

        t_Ω = AffineFETerm(a_i,b_i,trian,quad)
        op = AffineFEOperator(W,V,t_Ω)
        I_sol = solve(op)

        t_Ω = AffineFETerm(a_r,b_r,trian,quad)
        op = AffineFEOperator(W,V,t_Ω)
        R_sol = solve(op)

        if i%t_n_f ==0
            print(i)
            U[cont,1] = 1*S_sol
            U[cont,2] = 1*I_sol
            U[cont,3] = 1*R_sol
            cont +=1
        end

        S0 = S_sol
        I0 = I_sol
        R0 = R_sol
    end

end
```

```

    return U
end

#Guardamos y representamos los tiempos

t_n_euler = N_intervalos*t_n_f

tiempos = Array{Float64,2}(undef, 5,2)
for i=1:5
    t_n_f = 200 + (i-1)*100
    t_n_euler = N_intervalos*t_n_f
    t1 = @elapsed EulerMEF2(t_init,t_end,t_n_euler,t_n_f,U0)
    t2 = @elapsed EDP_parareal()
    tiempos[i,1] = t1
    tiempos[i,2] = t2
end

x_tiempos = LinRange(200, 600, 5);

plot(x_tiempos, tiempos, label = ["Euler" "Parareal"])

#Calculamos los errores cometidos y guardamos en ficheros para Paraview

U_EULER = EulerMEF2(t_init,t_end,t_n_euler,t_n_f,U0)

error = U_EULER - U_SIR[:,end,:]

for i=1:N_intervalos+1
    errS = error[i,1]
    writevtk(trian,"ErrorSuscep$(lpad(i,2,'0'))",cellfields=["error_S" => errS])
end

for i=1:N_intervalos+1
    errI = error[i,2]
    writevtk(trian,"ErrorInfec$(lpad(i,2,'0'))",cellfields=["error_I" => errI])
end

for i=1:N_intervalos+1
    errR = error[i,3]
    writevtk(trian,"ErrorRecup$(lpad(i,2,'0'))",cellfields=["error_R" => errR])
end

#Calculamos ahora los errores en norma L2

L2(w) = w*w

error_s = error[:,1];
error_i = error[:,2];
error_r = error[:,3];

error_S_L2 = [sqrt(sum( integrate(L2(i),trian,quad) )) for i in error_s];
error_I_L2 = [sqrt(sum( integrate(L2(i),trian,quad) )) for i in error_i];
error_R_L2 = [sqrt(sum( integrate(L2(i),trian,quad) )) for i in error_r];

using Plots

```

## A. CÓDIGO JULIA

---

```
y = [error_S_L2,error_I_L2,error_R_L2]
plot(T, y)

maximum(error_R_L2)

maximum(error_S_L2)

maximum(error_I_L2)
```



# Bibliografía

- [1] Jörg Nievergelt. Parallel methods for integrating ordinary differential equations. *Communications of the ACM*, 7(12):731–733, December 1964. 1, 7, 25
- [2] Alfio Maria Quarteroni and Alberto Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999. 2
- [3] Jacques-Louis Lions, Yvon Maday, and Gabriel Turinici. Résolution d’EDP par un schéma en temps < pararéel >. *Comptes rendus de l’Académie des sciences. Série I, Mathématique*, 332(7):661–668, 2001. Elsevier. 3, 33, 41
- [4] Francesc Verdugo and Santiago Badia. A user-guide to Gridap – grid-based approximation of partial differential equations in Julia. *arXiv:1910.01412 [cs]*, April 2020. 4
- [5] Efstratios Gallopoulos, Bernard Philippe, and Ahmed H. Sameh. *Parallelism in Matrix Computations*. Scientific Computation. Springer Netherlands, 2016. 6
- [6] Anna Doubova and Francisco Guillén González. *Un curso de cálculo numérico. Interpolación, aproximación, integración y resolución de ecuaciones diferenciales*. Universidad de Sevilla, 2007. 10
- [7] Michel Crouzeix and Alain L. Mignot. *Analyse numérique des équations différentielles*. Masson. 1984. 10
- [8] Grégoire Allaire. *Numerical Analysis and Optimization: An Introduction to Mathematical Modelling and Numerical Simulation*. OUP Oxford, May 2007. 11, 14, 18, 19, 20

## BIBLIOGRAFÍA

---

- [9] Philippe G. Ciarlet. *The finite element method for elliptic problems*. Number 40 in Classics in applied mathematics. Society for Industrial and Applied Mathematics, Philadelphia, 2002. 11
- [10] *The mathematical theory of finite element methods*. New York. 11, 12, 20
- [11] Alexandre Ern and Jean-Luc Guermond. *Theory and Practice of Finite Elements*. Springer Science & Business Media, March 2013. 11
- [12] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer New York, New York, 2010. 13, 14, 17, 18
- [13] Herbert B. Keller. *Numerical Solution of Two Point Boundary Value Problems*. SIAM, January 1976. 23
- [14] Martin J. Gander. 50 Years of Time Parallel Time Integration. In Thomas Carraro, Michael Geiger, Stefan Körkel, and Rolf Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, volume 9, pages 69–113. Springer International Publishing, Cham, 2015. 25
- [15] Martin Kiehl. Parallel multiple shooting for the solution of initial value problems. *Parallel Computing*, 20(3):275 – 295, 1994. 25
- [16] Martin J. Gander. Time Parallel Time Integration. November 2018. 33
- [17] Yvon Maday. The 'Parareal in Time' Algorithm. In F. Magoulès, editor, *Computational Science, Engineering & Technology Series*, volume 24, pages 19–44. Saxe-Coburg Publications, Stirlingshire, UK, May 2010. 33
- [18] William Ogilvy Kermack, Anderson Gray McKendrick, and Gilbert Thomas Walker. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, August 1927. 47
- [19] Fred Brauer. The Kermack-McKendrick epidemic model revisited. *Mathematical Biosciences*, 198(2):119–131, December 2005. 47

## BIBLIOGRAFÍA

---

- [20] Settapat Chinviriyasit and Wirawan Chinviriyasit. Numerical modelling of an SIR epidemic model with diffusion. *Applied Mathematics and Computation*, 216(2):395–409, March 2010. 53